Learning Java through Alice 3
4th Edition

- An Introduction to Programming

Tebring Daly and Eileen Wrigley

Table of Contents

Preface.....**Page 3**
- Acknowledgements
- About the Authors
- Approach
- Chapter Breakdown
- Organization
- Installing the Java and NetBeans
- Installing the Alice Environment
- Setting up NetBeans to Work with Alice

Chapter 0: Getting Started.....**Page 13**
- 0-1 What is Alice?
  - Alice Scene Setup
- 0-2 Hands-on Exercises
  - Exercise 1: Eat your Veggies Alice Program
  - Exercise 2: Alice in Wonderland Tea Party
- 0-3 Summary
- 0-4 Review Questions
- 0-5 Assignments

Chapter 1: Coding Introduction.....**Page 51**
- 1-1 What is Alice?
  - Alice Scene Setup
- 1-2 Introduction to Programming
- 1-3 What is Java?
  - History of Java
  - Java Capabilities
- 1-4 Programming Process
- 1-5 Documentation
- 1-6 Program Errors
- 1-7 Java Basics
  - Statements
  - Escape Codes
- 1-8 Hands-on Exercises
  - Exercise 1: Compiling and Executing a Java Program
  - Exercise 2: Drawing a Triangle Shape
- 1-9 Summary
- 1-10 Review Questions
- 1-11 Assignments

Chapter 2: Variables.....**Page 85**

2-1 Java Variables
Naming Variables
Using Primitive Data Types
Declaring Variables
Assigning Values to Variables
String Variables
2-2 Java Arithmetic
Precedence Rules
Examples
Modulus Explained
Shorthand Assignment Operators
Casting Rules
Walk-Through
2-3 Hands-on Exercises
Exercise 1: Making an Alien Walk in Alice
Exercise 2: Converting Fahrenheit to Celsius with Java*
Exercise 3: Transferring Alice to NetBeans Arithmetic Program
Exercise 4: Using Modulus in Money Changer Java Program*
2-4 Summary
2-5 Review Questions
2-6 Assignments

Chapter 3: Input/Output.....**Page 127**

3-1 Data Input
Scanner Class
JOptionPane Class
3-2 Formatting Output
NumberFormat Class
DecimalFormat Class
3-3 Importing Packages and Classes
3-4 Hands-on Exercises
Exercise 1: Mad Libs User Input
Exercise 2: Changing an Alice Clock's Time
Exercise 3: Computing Celsius with Input and Output*
Exercise 4: Money Changer with Input and Output*
Exercise 5: Computing Tip Using Input and Output
3-5 Summary
3-6 Review Questions
3-7 Assignments

* *Ongoing project*

Chapter 4: Conditionals.....**Page 161**

4-1 Conditional Execution
4-2 Alice Methods
4-3 Java Built-in Methods
Math Methods
String Methods
4-4 Java If Statements
Java Logical Operators
Java Hierarchy of Operators
4-5 Java Switch Statements
Hands-on Exercises
Exercise 1: Shark Moves to Closest Fish in Alice
Exercise 2: Determining the Tallest Object
Exercise 3: Display a Greeting Based on Time of Day
Exercise 4: Guessing Game
Exercise 5: Display a Message based on Temperature
Exercise 6: Validating a Password Entry
Exercise 7: Setting up a Simple Calculator
Exercise 8: Display a Random Quote using Switch Statement
4-6 Summary
4-7 Review Questions
4-8 Assignments

Chapter 5: Repetition.....**Page 209**

5-1 Loops
5-2 Alice Loops
While
For
5-3 Java Loops
While
Do/While
For
5-4 Nested Loops
Break Statement
5-5 Hands-on Exercises
Exercise 1: While Loop Tree Growing
Exercise 2: For Loop to Print 99 Bottles of Soda Song
Exercise 3: Tallying Coin Tosses
Exercise 4: Modifying Guessing Game to Use a Loop*
Exercise 5: Drawing a Grid using Nested Loops
5-6 Summary
5-7 Review Questions
5-8 Assignments

Chapter 6: Arrays.....**Page 245**

6-1 Declaring and Creating Arrays
Declaring an Array
Creating Array Objects
6-2 Accessing and Using Created Arrays
Accessing Array Elements
Length of Array
Two-Dimensional Arrays
6-3 Hands-on Exercises
Exercise 1: Entering Scores into an Array
Exercise 2: Using Command Line Args Box to Enter Scores
Exercise 3: Using JOptionPane to Enter Scores
Exercise 4: Using a Two-Dimensional Array to Enter Scores
Exercise 5: Finding the Distance Between Two Cities
Exercise 6: Bubble Sort
Exercise 7: Using a For Each loop for a Penguin Array
6-4 Summary
6-5 Review Questions
6-6 Assignments

Chapter 7: Procedural Methods.....**Page 287**

7-1 Introduction to Methods
7-2 Overview
Classes
Objects
Methods
7-3 Java Application Programming Interface (API)
7-4 Java Method Declaration
7-5 Java Method Examples
Methods with No Parameters
Methods with Some Parameters
7-6 Hands-on Exercises
Exercise 1: Creating a Stomp in Alice
Exercise 2: Making a Dog's Tail Wag in Alice
Exercise 3: Hokey Pokey in Alice*
Exercise 4: Creating a Moon Walk Dance in Alice
Exercise 5: Writing "Old MacDonald Had a Farm" Song
7-7 Summary
7-8 Review Questions
7-9 Assignments

* *Ongoing project*

Chapter 8: Functional Methods.....**Page 339**

8-1 Functional Methods Explained
8-2 Java Built-in Functional Methods
Java Application Programming Interface
Math Functions
String Functions
8-3 Alice Built-in Functional Methods
8-4 Method Declaration
8-5 Functional Method Example
8-6 Hands-on Exercises
Exercise 1: Using Alice Built-in Functional Methods
Exercise 2: Finding the Circumference of Alice UFOs
Exercise 3: Calculating BMI
Exercise 4: Writing Calculations Methods in Another Class
8-7 Summary
8-8 Review Questions
8-9 Assignments

Chapter 9: Classes and Objects.....**Page 367**

9-1 Object-Oriented Concepts
9-2 Java Classes
9-3 Setter and Getter Methods
Alice Setter and Getter Methods
Java Setter and Getter Methods
9-4 Visualizing Your Application with UML
9-5 Hands-on Exercises
Exercise 1: Practicing with Object Oriented Concepts
Exercise 2: Using Setters and Getters
Exercise 3: Adjusting the Hokey Pokey for All Bipeds*
9-6 Summary
9-7 Review Questions
9-8 Assignments

Chapter 10: Graphics.....**Page 407**

10-1 Introduction to Graphical User Interfaces
GUI Packages – AWT and Swing
Creating a Window
Placing GUI Components in the Window
10-2 Graphics
Drawing
Layout
Changing the Font
Changing the Color
Printing Strings
Drawing Lines

Drawing Rectangles
Drawing Ovals
Drawing Arcs
10-3 Using Paint to Determine Pixel Locations
10-4 Hands-on Exercises
Exercise 1: Drawing a Happy Face
Exercise 2: Drawing a Car
Exercise 3: For Loop to Print Piano
Exercise 4: For Loop to Print Checkerboard
Exercise 5: For Loop to Animate Text
Exercise 6: Drawing Polygons
10-5 Summary
10-6 Review Questions
10-7 Assignments

Comprehensive Project.....**Page 449**

Appendix.....**Page 511**
Exception Handling
NetBeans Debugger

Acknowledgments

First and foremost, the authors would like to convey special thanks to the Alice Software team at Carnegie Mellon University for providing the Alice software to make this text possible. Also, we would like to take this opportunity to thank Electronic Arts, Inc. for providing their rich set of graphics which certainly makes Java and Alice programming more interesting. In addition, we would like to express our gratitude to the National Science Foundation and the Alice ATE grant team members for their continued support. Words cannot express our gratitude to Wanda Dann who was abundantly helpful and offered invaluable assistance, encouragement, and guidance.

A heartfelt thanks to those that helped with revisions to this book: Bob Benavides (Computer Science Professor at Collin College) and Lisa Moran (Student at Collin College).

About the Authors

This book has been a joint effort by a mother and daughter team.

Eileen Wrigley, full-time Professor of Computer Information Technology courses at the Community College of Allegheny County in Pittsburgh, Pennsylvania brings more than 40 years of teaching experience to her writing. She earned her B.S. and M.S. degrees from the University of Pittsburgh in Mathematics and Computer Science.

Dr. Tebring Daly has been teaching full-time in the Computer Science department at Collin College in Plano, Texas since 2006. She has earned her B.S. and M.S. degrees from the University of Pittsburgh and Ph.D. from the University of North Texas.

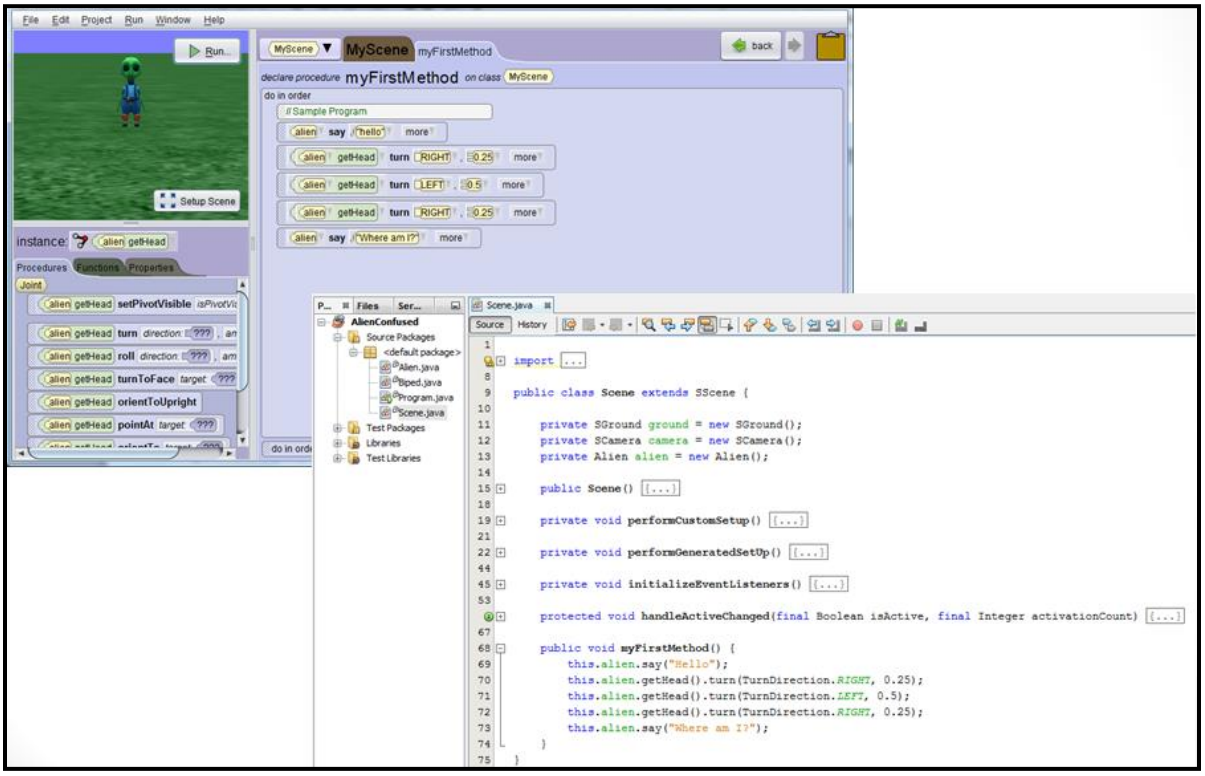
Approach

This book is designed for students wanting to learn fundamental programming concepts. No previous programming experience is required. All of the software used in this text are available to download free of charge. The versions of the software may differ slightly from the versions used in this text since the versions are constantly being updated.

This book will teach you how to program by using Java code. We will use a Java editing tool called NetBeans[1] to help write the code. The environment can be used on a Windows® operating system[2], Apple Macintosh® operating system[3] (Mac), or Linux® operating system[4].

[1] Supported by Oracle, http://netbeans.org
[2] Microsoft Corporation, http://www.microsoft.com
[3] Apple Macintosh Corporation, http://www.apple.com/osx
[4] Linux Foundation, http://www.linux.org

We are also using a tool called Alice 3[5], to provide visuals for abstract programming concepts. This environment works on Windows, Mac, and Linux operating systems. Alice 3 is a drag and drop environment that can be transferred into Java code in the NetBeans environment as shown below.



Chapter Breakdown

Preface provides an overview of the text and the instructions for downloading the required software.

Chapter 0 will introduce you to the Alice 3 environment.

Chapter 1 describes the history of Java, basic programming terminology, and writing Java code in NetBeans.

Chapter 2 covers naming rules, creating and using variables in code, using arithmetic statements, order of operation, shorthand operators, and casting rules.

Chapter 3 explains various ways of formatting output and receiving user inputs. The user is introduced to import statements.

[5] Developed by Carnegie Mellon University (CMU), Alice, http://www.alice.org

Chapter 4 provides an explanation and practice with relational and logical operators using conditionals.

Chapter 5 shows three types of repetition techniques (while, do while, and for loop).

Chapter 6 explains how to use an array to store multiple values of the same type.

Chapter 7 shows the user how to modularize programs using methods. The user is introduced to the Java documentation and the syntax for writing procedural methods.

Chapter 8 expands upon chapter 7 to include methods that return values.

Chapter 9 talks about object-oriented terms (encapsulation, inheritance, and polymorphism) and their use.

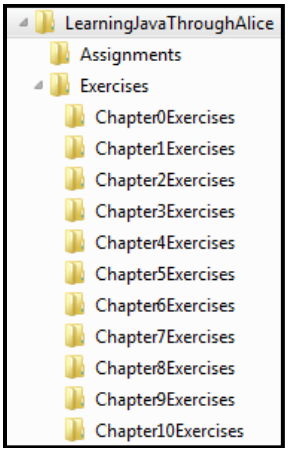
Chapter 10 provides an introduction to GUI and the structure for creating basic drawings.

Organization

Each chapter is divided into content segments, hands-on exercises, a summary, and review questions. You should work through the hands-on exercises in each chapter.

The data files needed for the hands-on exercises and assignments can be found at http://faculty.collin.edu/tdaly/book4/.

You may want to create an organizational method for keeping track of your files. Each chapter has several exercises that will walk you through the programming concepts for that chapter. There is at least one assignment at the end of every chapter. The assignments test your ability to put the concepts from the chapter into action on your own. Please save the chapter exercises to the "Exercises" folder and the assignments at the end of each chapter to the "Assignments" folder so that you don't get confused.



Instructors: Please email tdaly@collin.edu for solutions, sample syllabi, etc.

Installing the Java and NetBeans

Java is an object-oriented programming language. We will be writing all of our Java code in NetBeans. NetBeans is not the only environment for writing Java code, but it is what we will be using for this text.

You should download the NetBeans and Java SDK (Software Development Kit) bundle. This bundle will include everything that you will need to write and run Java programs. Please follow the install directions located on the following website: http://faculty.collin.edu/tdaly/book4/

Installing the Alice Environment

Alice 3 provides a 3D environment for manipulating objects using drag and drop code segments. This environment helps to provide visual representations of abstract programming concepts. Please follow the install directions located on the following website: http://faculty.collin.edu/tdaly/book4/

Setting up NetBeans to Work with Alice

There is an Alice 3 plugin file that you will also need to download and add to the NetBeans environment. Please follow the install directions located at the following website: http://faculty.collin.edu/tdaly/book4/

Chapter 0

Getting Started

Objectives

- ☑ Explain the purpose of Alice
- ☑ Setup an Alice scene
- ☑ Code an Alice animation

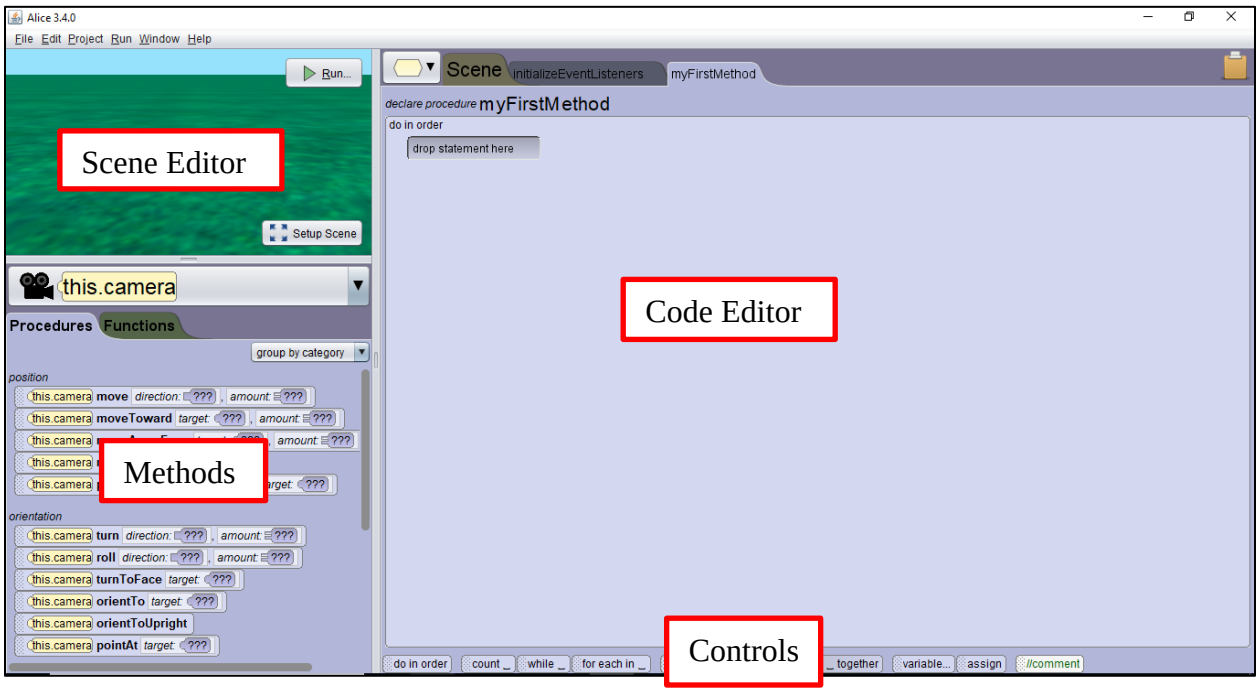
What is Alice?

The Alice team at Carnegie Mellon University named the Alice programming software in honor of Lewis Carroll who wrote Alice's Adventures in Wonderland. Lewis Carroll was able to do complex mathematics and logic, but he knew that the most important thing was to make things simple and fascinating to a learner.

Alice makes it easy to create an animation or interactive game. It is designed for beginners who want to learn object-oriented programming. In Alice, 3-D objects (e.g., people, aliens, animals, props) are placed in a scene. Then, students drag and drop tiles to create a program to animate the objects. These tiles correspond closely to statements in Java. Alice allows students to immediately see how their programs run, enabling them to easily understand the relationship between the programming statements and the behavior of objects in their program.

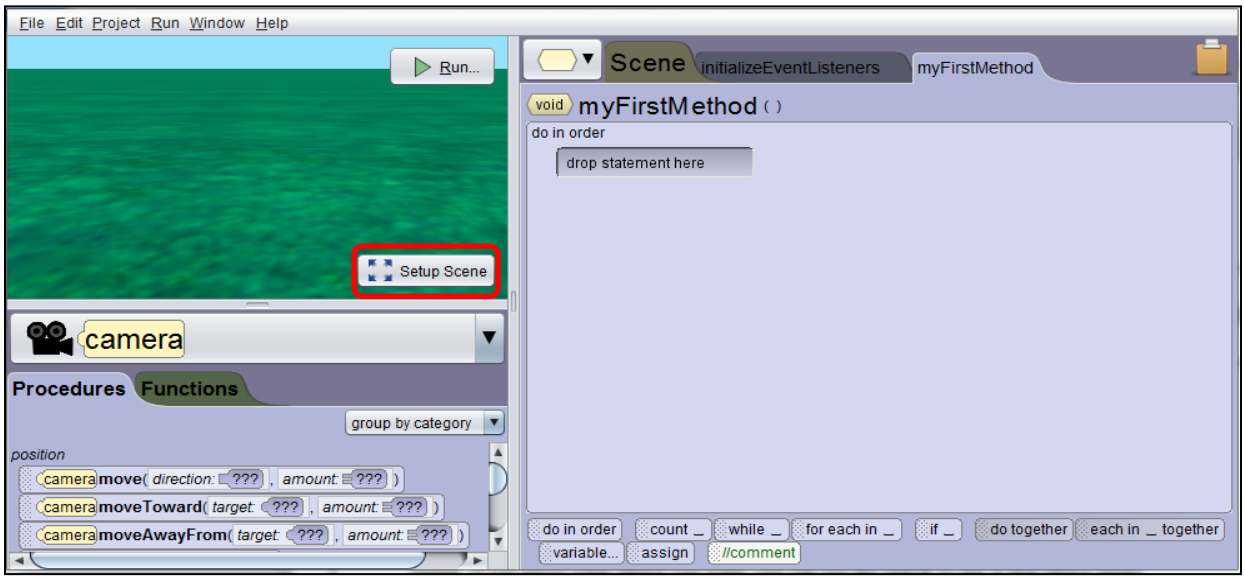
Alice 3 is the newest version of the Alice software. This version of the software allows users to transfer Alice projects into the NetBeans environment to edit the Java code. This text will be using Alice to demonstrate fundamental programming concepts in Java such as objects, methods, looping, etc. by creating animations.

An Alice scene begins with a template for an initial scene. These templates can be grass, water, snow, etc. Then, you add various objects to the scene to create the virtual scene that you desire.

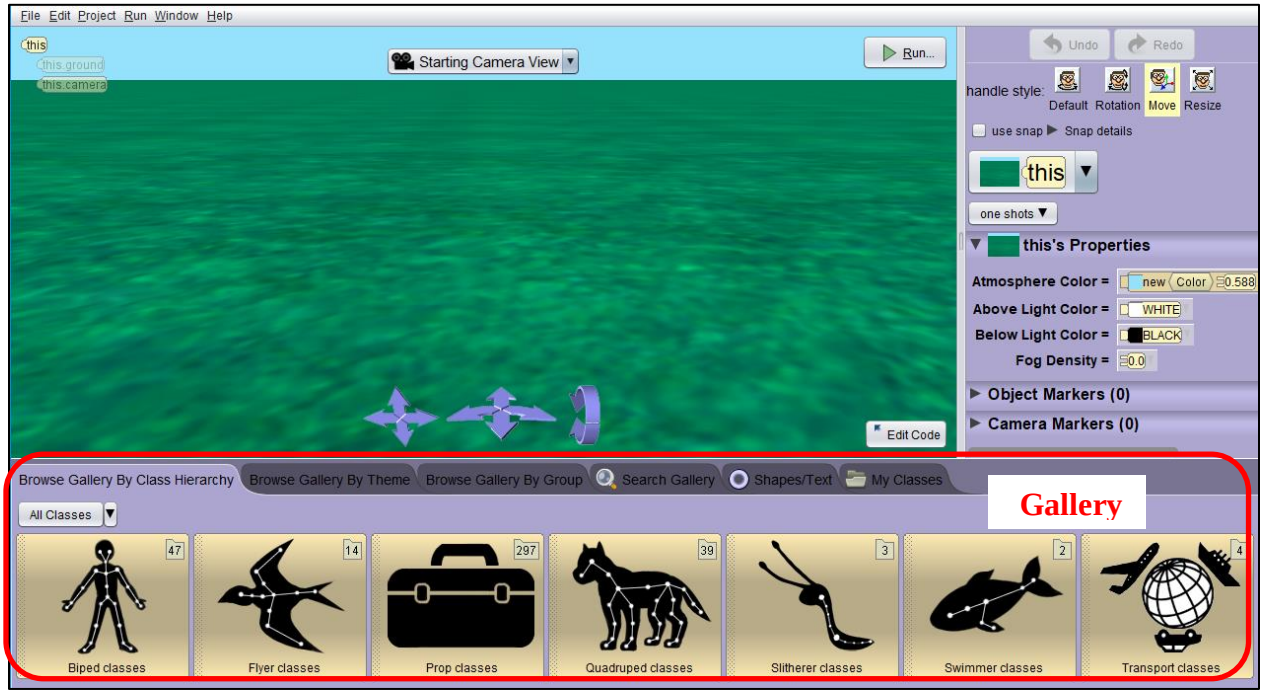


Alice Scene Setup

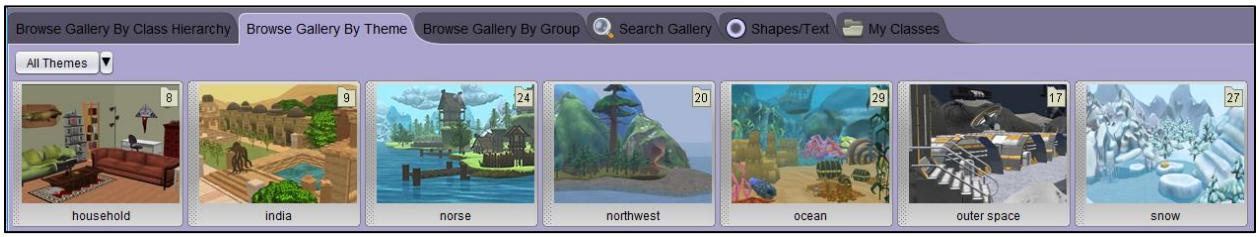
Objects are added to the scene via the scene editor (Click on Setup Scene button).



There are several choices for selecting objects from the gallery. The **hierarchy** choice is broken down by physical makeup. A biped has 2 legs, a flyer has wings, a prop is something that is inanimate, a quadruped has 4 legs, and a swimmer has fins.



You can also view the objects by **theme** or by **group** as shown below. The **search** feature is nice if you are looking for a particular object.



The Alice developers have provided a number of 3D models for you to use in your animations. **An Alice 3D model (class) is a blueprint that tells Alice how to create a new object in the scene.** The 3D model provides instructions on how to draw the object, what color it should be, what parts it should have, its size (height, width, and depth), and many other details. Once you decide what objects, you would like to have, you will need to click on the class to create an object of that type. For example, if I want a girl object in my world, I would select the Biped folder and then the Adult class to create the girl object.

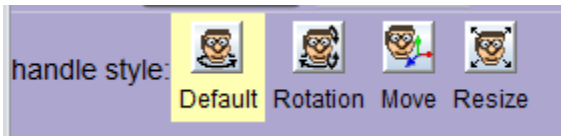
When you create an object, you will need to give it a name. You can leave the default name or give it your own name. You cannot give two objects the same name. Be careful when you are creating objects, if you try and use the name girl more than once, it won't let you create the new object.

All objects will initially be placed into the middle of the scene and then can be manipulated to any position desired. Alternatively, objects can be dragged to any position in the scene.

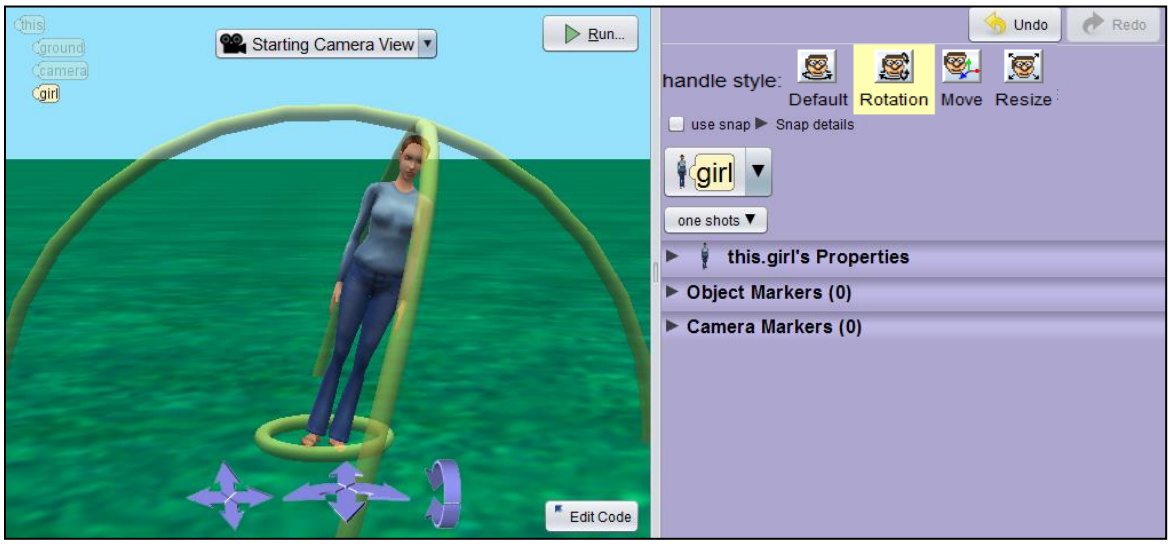
Alice objects are represented in a three-dimensional space. Each object has width, height, and depth as shown below. The height is measured vertically, the width is measured horizontally, and the depth is measured from front to back.

There are six possible directions in which an object may move – **forward, backward, up, down, left and right**. Remember that directions are left and right with respect to the object, not the camera's point of view. For example, this girl object can move forward, backward, up (in air), down (into ground), her left, or her right. The direction an object is facing and where the top of the object is located (relative to the world) is known as the **object's orientation**. In the scene editor, there are 4 buttons that allow you to manipulate the object.

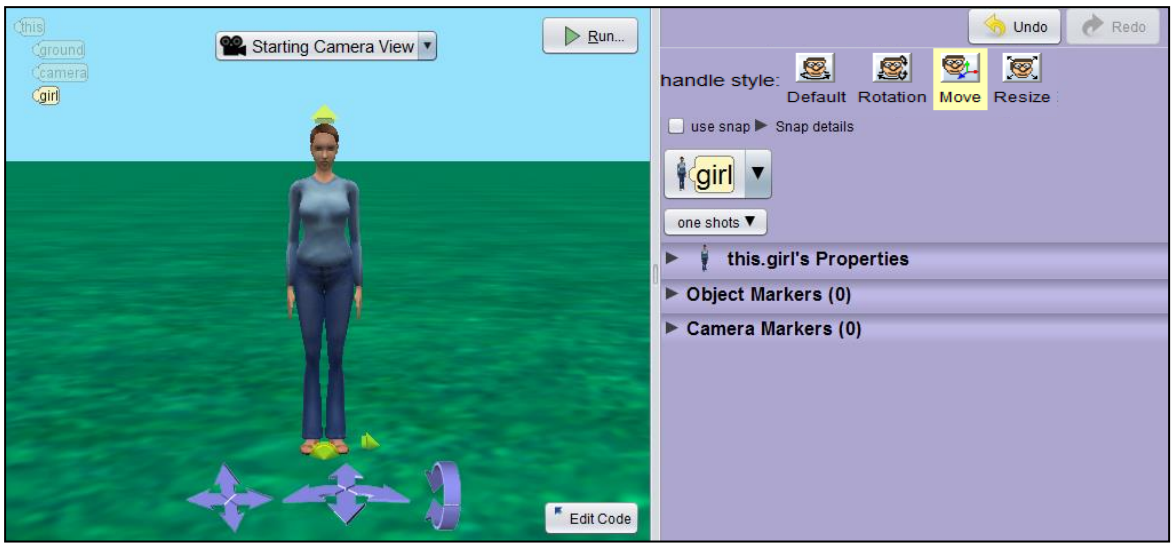
The **DEFAULT** button allows you to rotate the object. Hold down the left mouse button and drag the circle to rotate the object.



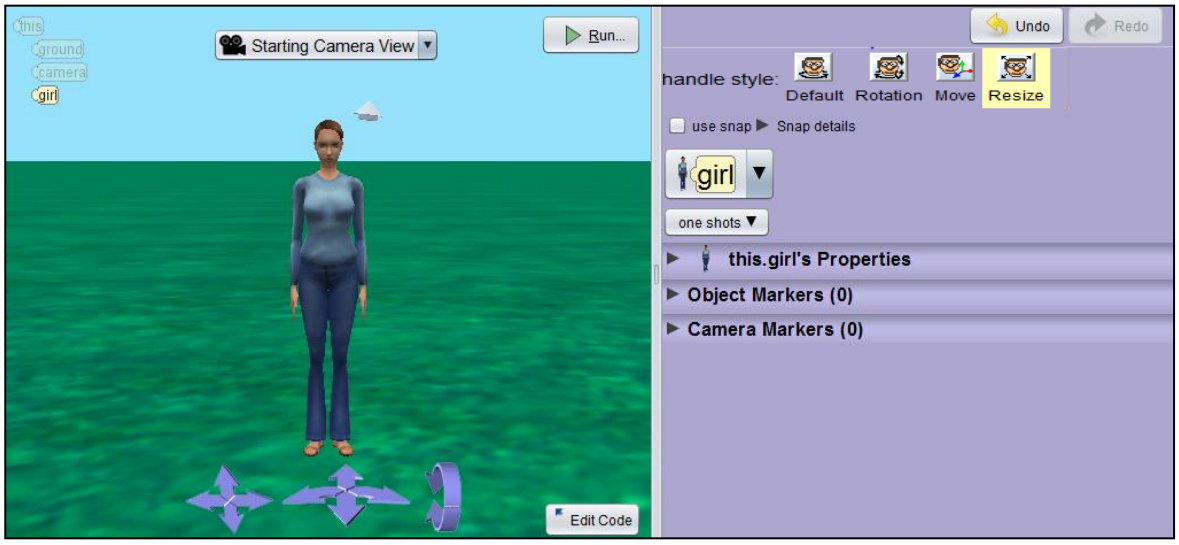
The **ROTATION** button allows you to do rotations in all directions. Hold down the left mouse button and drag the appropriate circle to rotate the object.



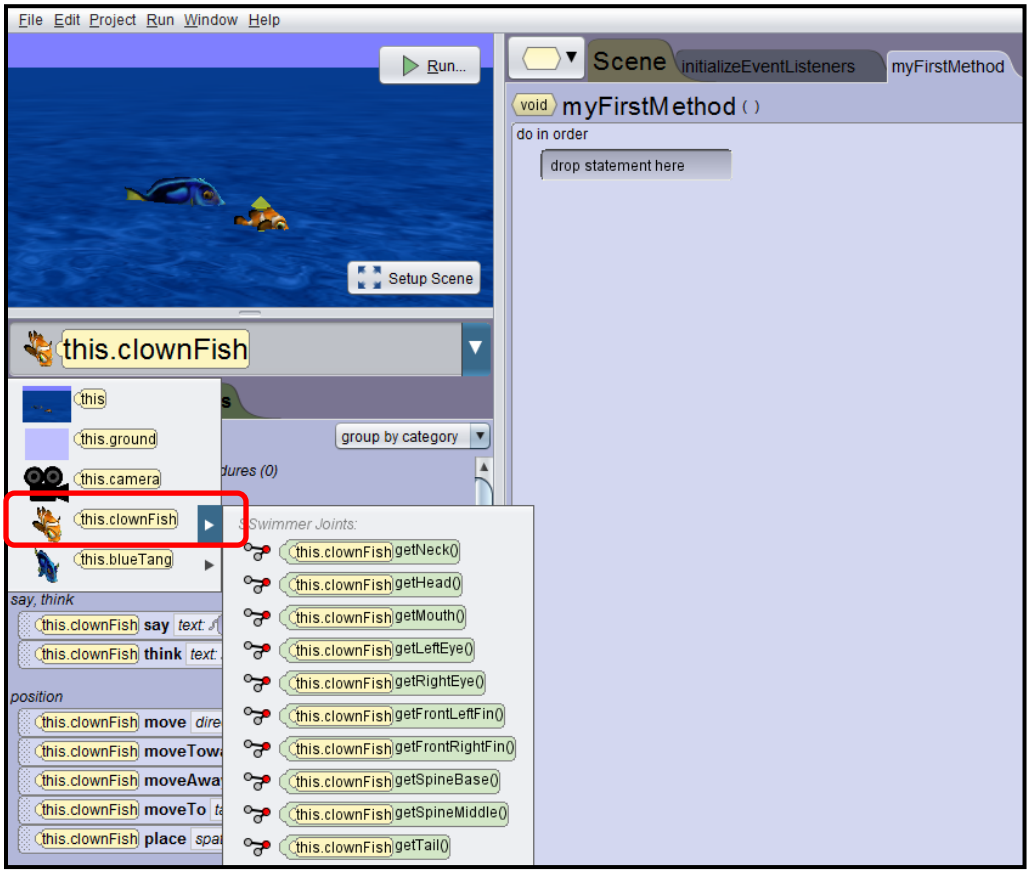
The **MOVE** button allows you to move the object in all directions. Hold down the left mouse button and drag the arrows to move the object. The arrow at the top of the object moves the object up and down, the arrow on the right of the object moves the object left and right, and the arrow in front of the object moves the object forward and backward.



The **RESIZE** button allows you to resize the object. Hold down the left mouse button and drag the arrow at the top of the object. The object will resize proportionately.



All of the Alice models have body parts that can be manipulated with rolls, turns, etc. You can access the subparts for an object by clicking the part drop down next to the object drop down.



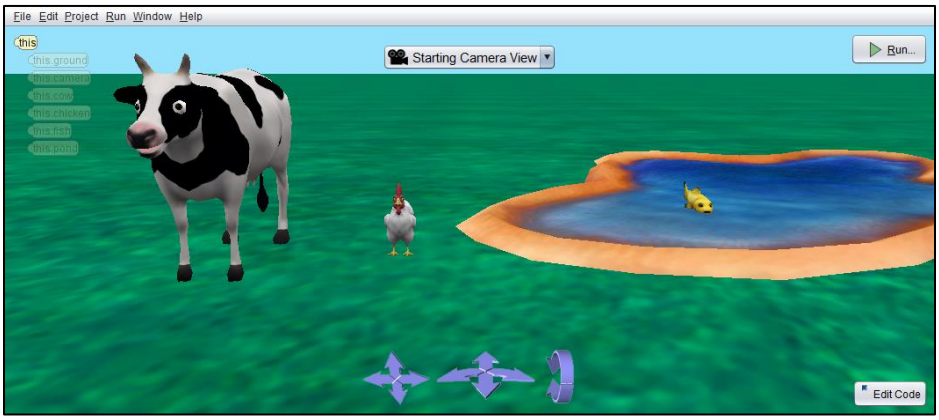
The best way to see how Alice 3 works is to create a virtual world with objects and animate the objects in that world. This will be done in the hands-on exercises.

Hands-on Exercises

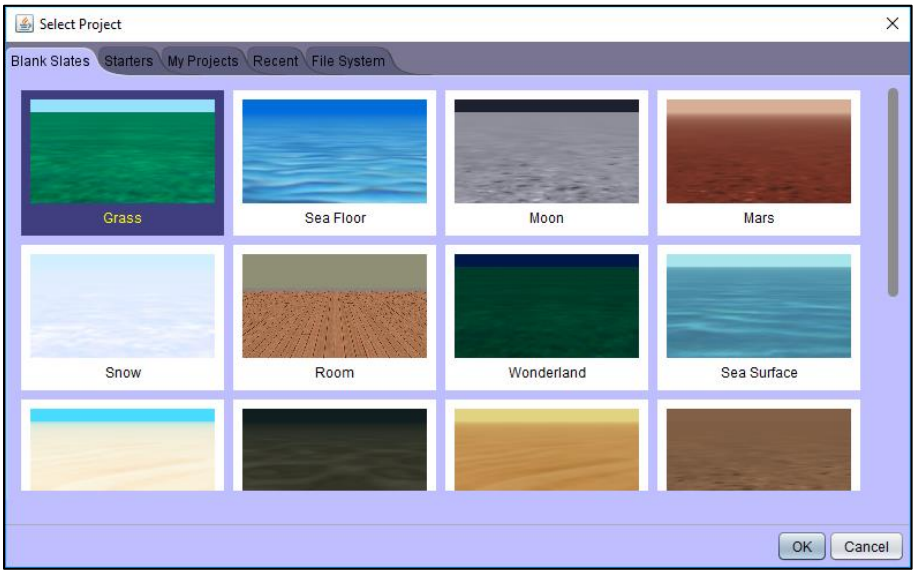
Exercise 1: Eat your Veggies Alice Program

1. Open up Alice 3. You will need to find the installed Alice 3 folder and double click on the **Alice 3** application file. The first step in programming is understanding the problem. We would like to create an animation that has the cow try to convince people to eat chickens, the chicken tries to convince the people to eat fish, and the fish tries to convince people to go vegan. Once you understand the problem, you will setup the scene and create a storyboard for animating the scene.

2. Our goal for the scene setup is to have the following animals: cow, chicken, and fish. When we are finished it should look similar to the following:

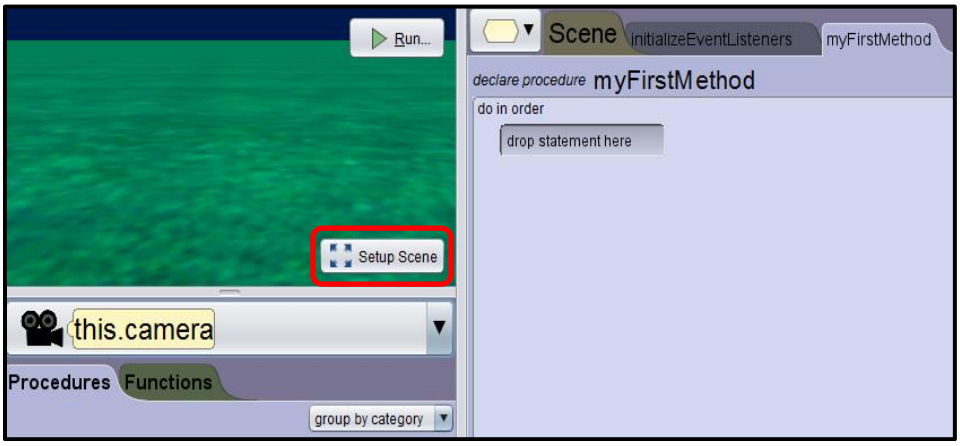


3. Select the **grass** template:



4. Select **File** from the menu, then **Save As**. Save this file as **Vegan**. Please get in the habit of capitalizing the first letter of every word in your filename and do not use spaces when naming your files. You should save your work often. You can click Save from the File menu from this point on. You should save this file in your **Chapter1Exercises** folder.

5. Click on **Setup Scene button.**

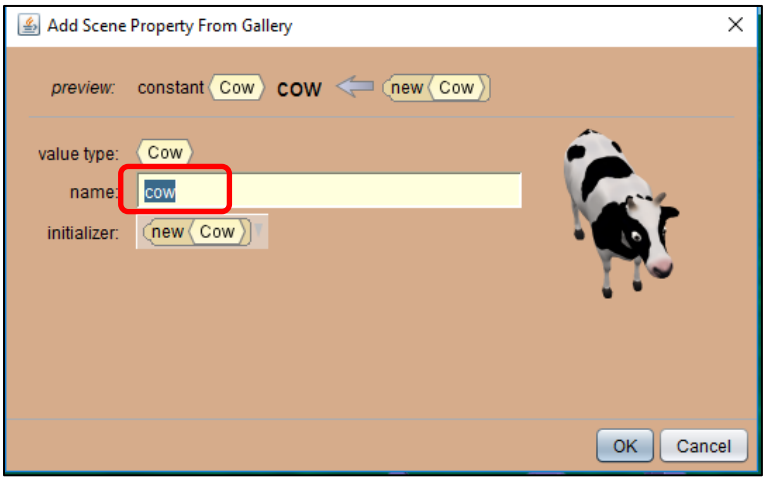


6. First we are going to add a cow. Click on the tab called **Browse Gallery By Group**. Click on the **animals** folder.

7. Scroll to the **Cow** class (they are in alphabetical order). You could have used the ***Search Gallery*** tab to find the cow as well.

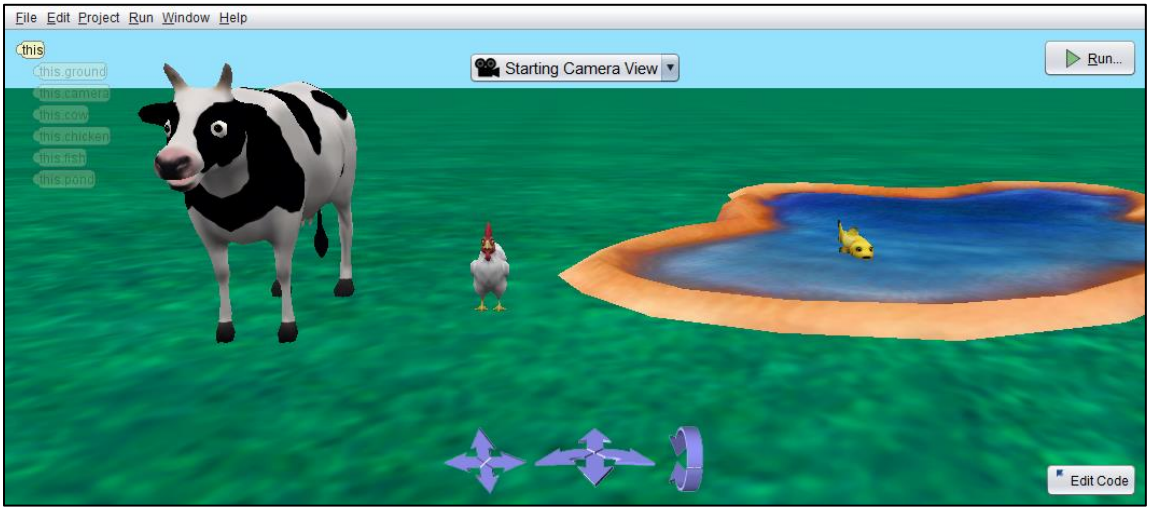
8. Click on the **Cow** class to add a cow to your world or hold down your left mouse button and drag this object to wherever you would like to place it in your scene. If you choose to click on the Cow class, the new object will be placed automatically in the center of the scene.

9. When you click on the class it will ask you for a name for the object. You can leave the name *cow* or rename if you want. Do not put spaces in your object name and the first letter of your object name should begin with a lowercase letter and the first letter of the second word should be a capital letter. Object names begin with a lowercase letter.



10. Please add a **chicken** in the same way that you added the cow. There will be several chickens to choose from, pick whichever one you like. Name this chicken whatever you would like, but be sure to follow the naming rules. Do not put spaces in your object name and the first letter of your object name should begin with a lowercase letter and the first letter of the second word should be a capital letter. Object names begin with a lowercase letter.

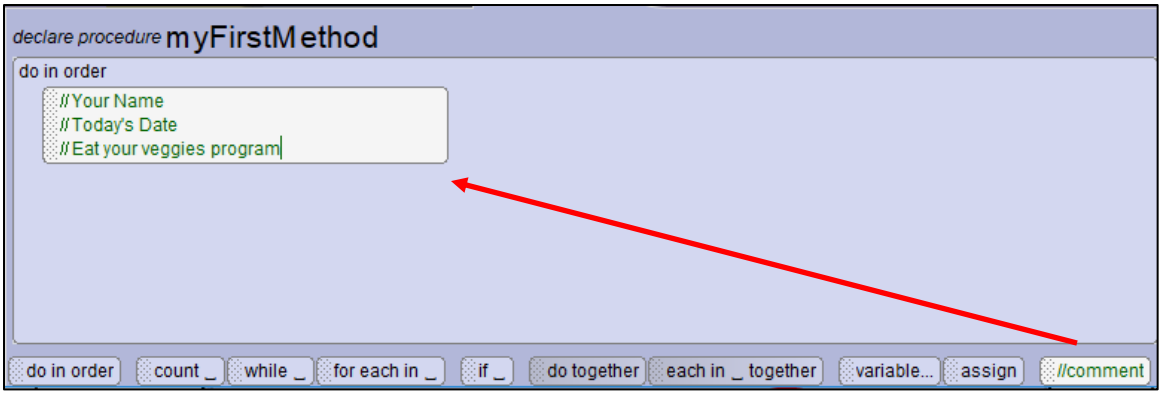
11. Next, we are going to add a **fish**. Now we can test out the search feature in Alice by typing **fish** into the search box. Scroll to the end to see the fish. Please choose the fish that you like. You can search for a **pond** to add to the environment as well. Place the fish in the pond.



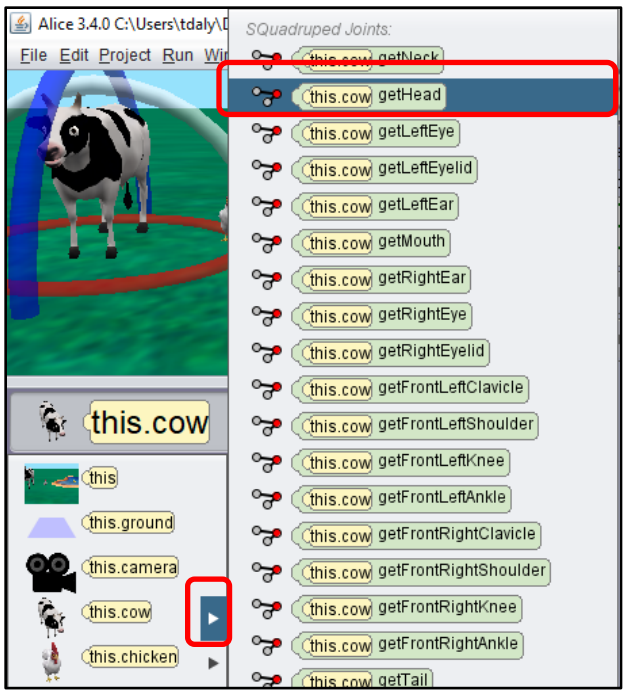
12. We are finished with the scene setup.
13. Before writing the code for our animation, we should first create a storyboard of what we wish to accomplish.

- Scene opens with the cow eating
- Cow says “Eat more chicken”
- Chicken moves forward
- Chicken says “Eat more fish”
- Fish says “Go vegan”
- Fish hides under the water

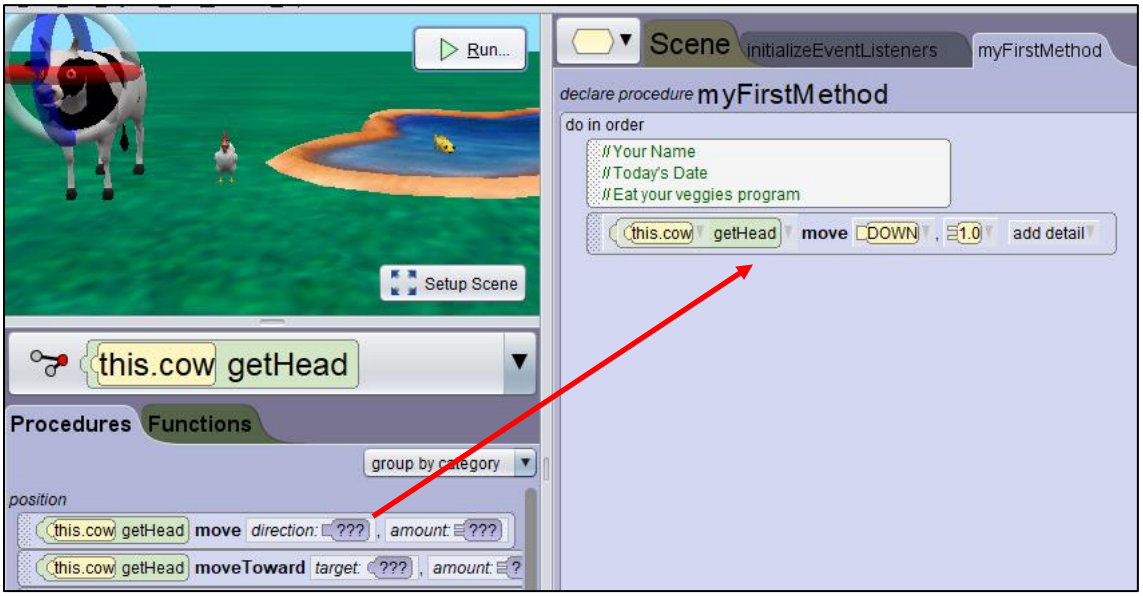
14. Drag the **//comment** block to the editor and enter your comments. You need to put your name, the date, and a description of the program in comments at the top of all of your programs.



15. Select the cow's head by selecting the **cow** from the object drop down and then clicking on the arrow to the right to display the subpart menu. Choose the **this.cow.getHead** from the list.

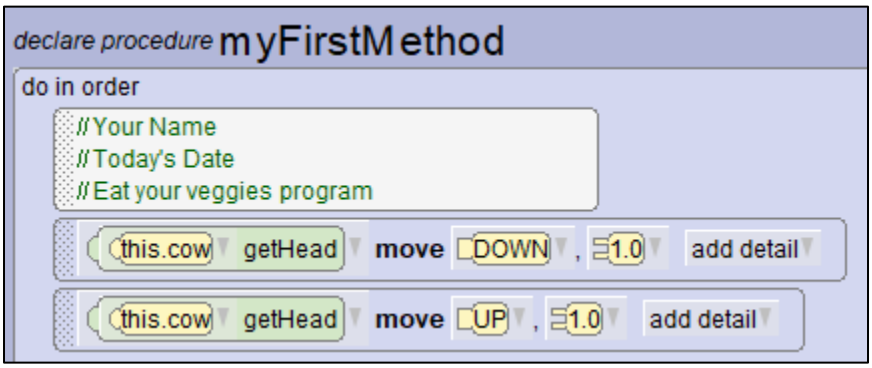


16. Select the **move** method and drag it to the scene editor as shown below. Select **DOWN** as the direction argument and **1.0** as the amount argument. A method is an action that an object can do and an argument answers a question. The DOWN argument answers the question of what direction the cow's head should move. The 1.0 argument answers the question of how much the cow should move its head.

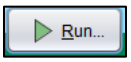


17. Test your program by clicking on the **Run** button.

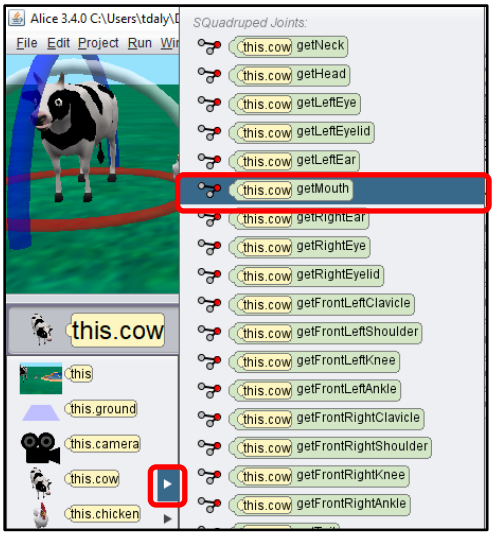
18. Now, let's make the cow put his head back up, by selecting the **move** method for the cow's head again, but this time choosing **UP** as the direction argument and **1.0** as the amount argument.



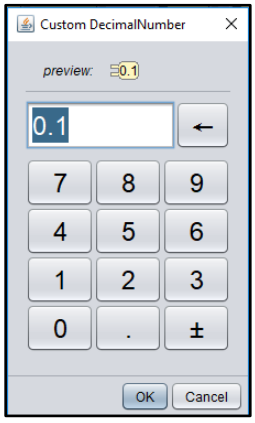
19. Test your program by clicking on the **Run** button.



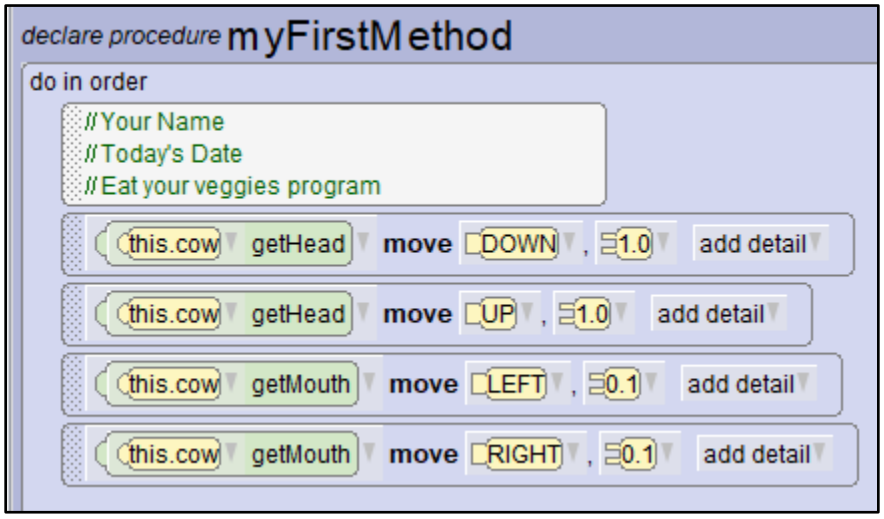
20. Next, let's have the cow move his mouth left and right. We will need to click on the arrow next to the cow object to select the cow's mouth.



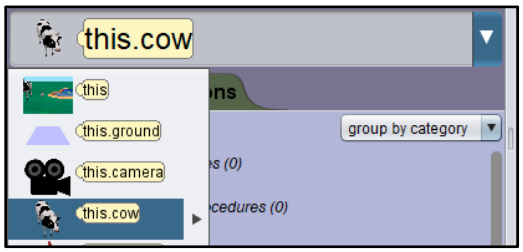
21. Now, drag the move method for the cow's mouth to the scene editor. Select **LEFT** as the first argument that answers the question of what direction you want the cow's mouth to move and select **0.1** as the second argument that answers the question of how much you want his mouth to move. You will notice the 0.1 is not an option in the drop down list for the amount. You will need to select **Custom DecimalNumber…** and then type 0.1 in.



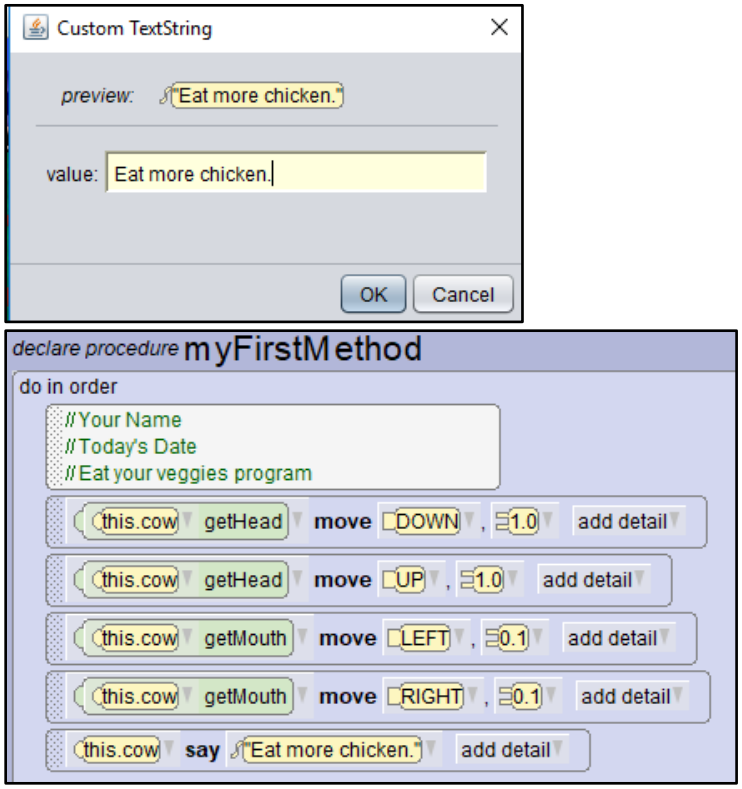
22. Your code should look as follows. Run you program to see the results thus far. It should look like the cow is eating grass.



23. Next, let's have the cow say "Eat more chicken". You will first need to select the whole cow object from the list.

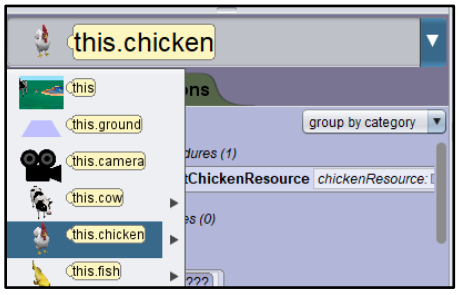


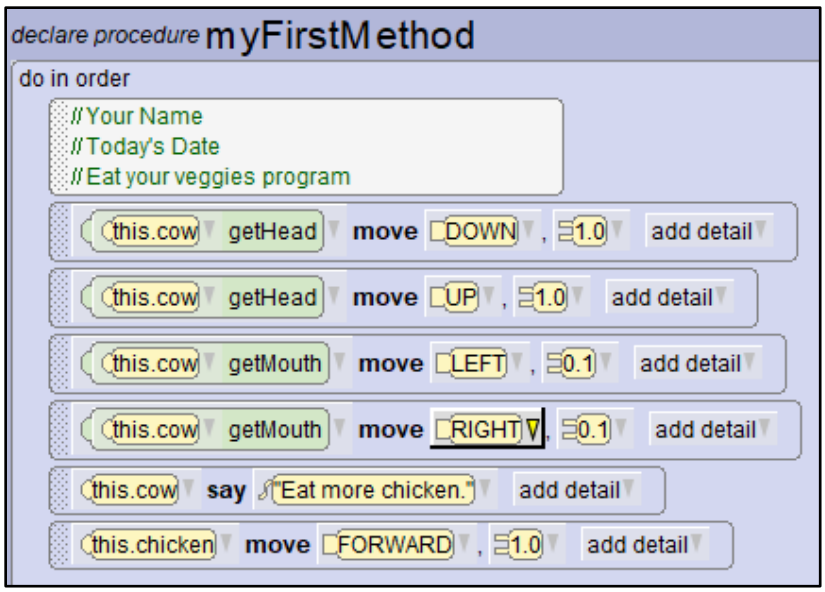
24. Then, drag the **say** method over to the code editor and select **Custom TextString.** Type in the following string: "**Eat more chicken."** into the text box.



25. Run your program to test it.

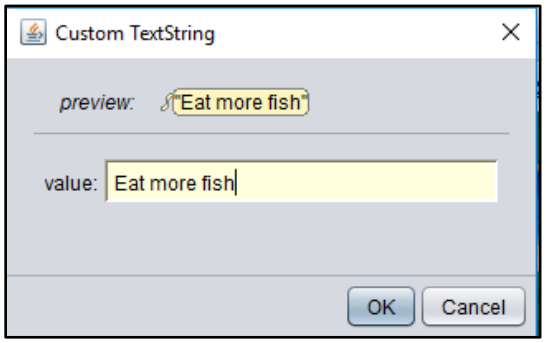
26. Next, let's select the whole chicken and drag the move method into the code editor for the chicken. We will select **FORWARD** as the first argument and **1.0** as the second argument.





27. Run your program. Does the chicken move forward?

28. Next, drag the **say** method for the chicken to the code editor and select **Custom TextString.** Type in the following string: "**Eat more fish."**

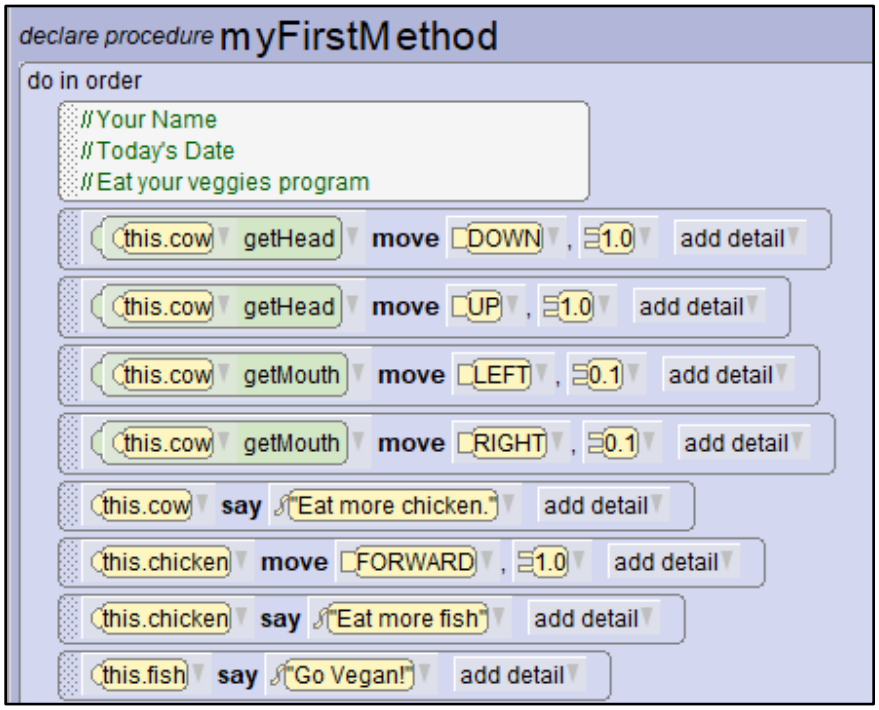


29. Run your program to see your animation.

30. Finally, have the fish respond to the cow and chicken, by telling the user to go vegan so they won't cause any animal suffering. Click on the whole fish, then drag the say method to the code editor and select Custom TextString. Type in the following string: "**Go Vegan!"**

31. Next, let's select the whole fish object and drag the move method into the code editor for the fish. We will select **DOWN** as the first argument and **1.0** as the second argument. This will make the fish look like he is going under water to escape being eaten.

32. Your final code should looks as follows.



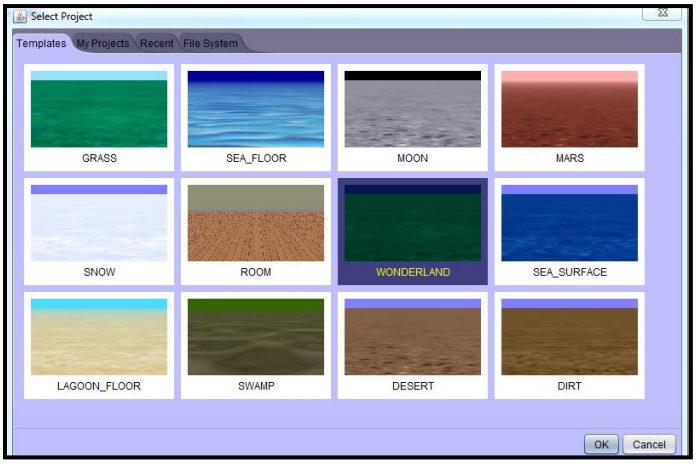
33. Run your program to test your animation. Save your work and exit Alice.

Exercise 2: Alice in Wonderland Tea Party

1. Open up Alice 3. You will need to find the installed Alice 3 folder and double click on the **Alice 3** application file. The first step in programming is understanding the problem. We would like to create a trimmed version of the Alice in Wonderland unbirthday tea party. Once you understand the problem, you will setup the scene and create a storyboard for animating the scene.

2. Our goal for the scene setup is to have the following characters: Alice, Mad Hatter, and the March Hare. We will also add some objects to make the scene more interesting: a table, chairs, a tea pot, tea cups, and a birthday cake. When we are finished it should look similar to the following:

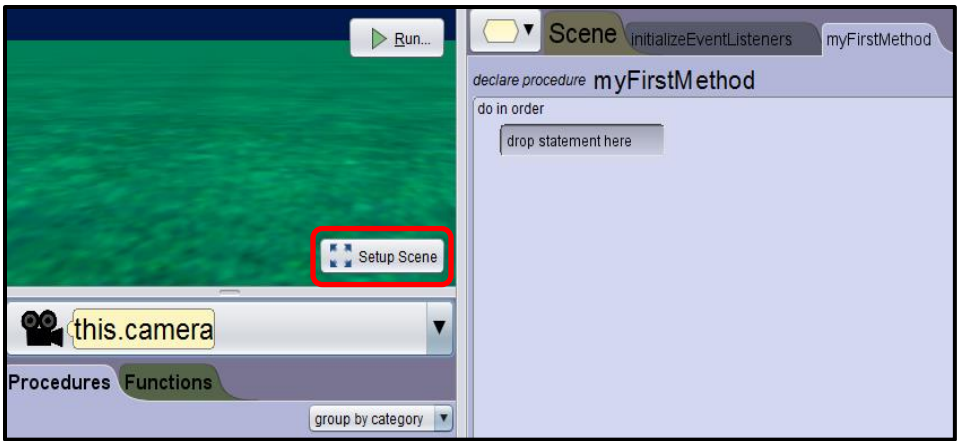
3. Select the **wonderland** template:



4. Select **File** from the menu, then **Save As**. Save this file as **TeaParty**. Please get in the habit of capitalizing the first letter of every word in your filename and do not use spaces when naming your files. You should save your work often. You can click Save from the

File menu from this point on. You should save this file in your **Chapter1Exercises** folder.

5. Click on **Setup Scene button.**



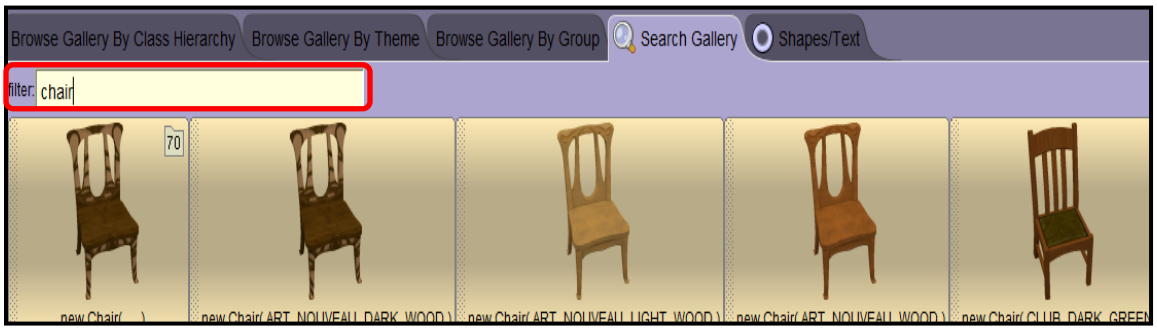
6. We are going to add a table to the scene for the characters to gather around. There is a tea table specifically designed for Alice in Wonderland. Click on the tab called **Browse Gallery By Class Hierarchy**. Click on the **Prop classes** category.

7. Scroll to the end (they are in alphabetical order) until you see the **TeaTable** class. You could have used the ***Search Gallery*** tab to find the table as well.

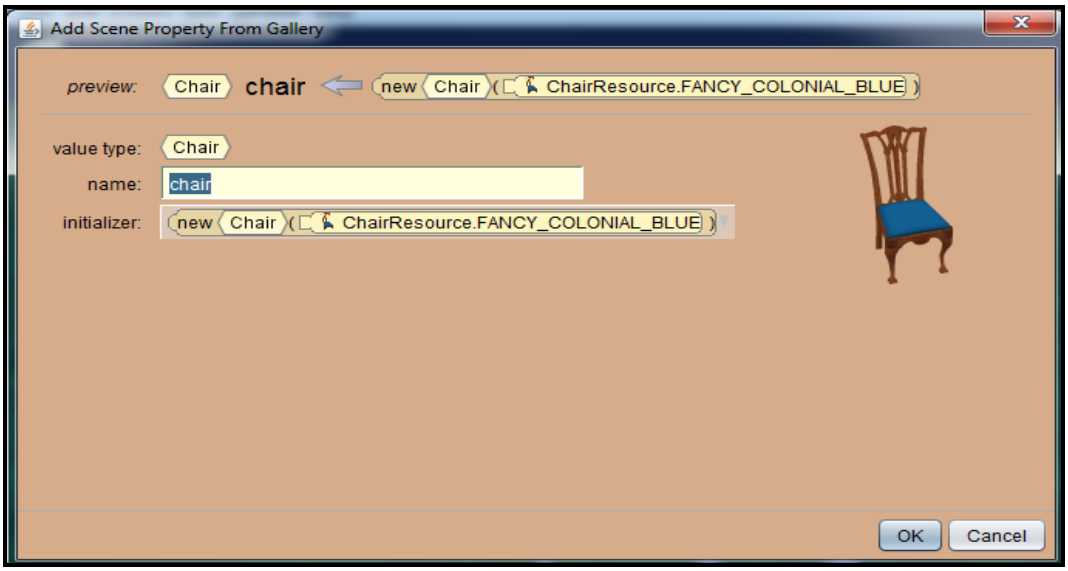
8. Click on the **TeaTable** class to add a tea table to your world or hold down your left mouse button and drag this object to wherever you would like to place it in your scene. If you choose to click on the TeaTable class, the new object will be placed automatically in the center of the scene.

9. When you click on the class it will ask you for a name for the object. You can leave the name teaTable or rename if you want. Do not put spaces in your object name and the first letter of your object name should begin with a lowercase letter and the first letter of the second word should be a capital letter. Object names begin with a lowercase letter; we will talk more about this in the next chapter.

10. Next, we are going to **add a chair**. Now we can test out the search feature in Alice by typing chair into the search box. You will have a list of all the chair models. Please choose the chair that you like.

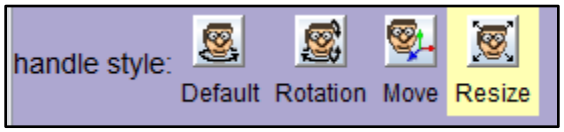


11. Drag the chair that you want onto the scene where you want it by holding down the left mouse button and dragging from the class that you are choosing to add. You will see a yellow bounding box that shows you were your new object will be placed. When you get the object where you want it, release and it will ask you for a name for the object.



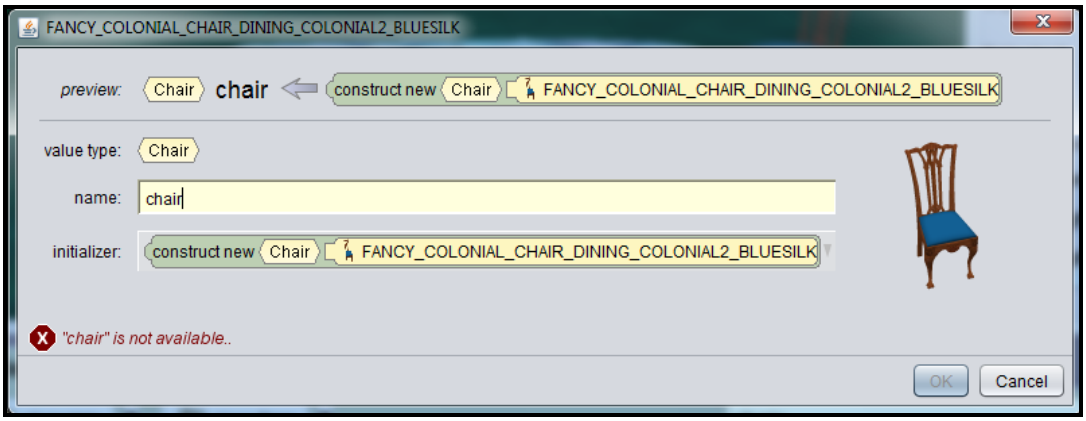
You should name this object something simple. Let's name it **chair**.

12. We should resize the chair so that it matches the size of the table. To do this, you will need to select the chair and then click on the **resize** button from the handle style choices. When you click on the button, an arrow will appear above the chair. Holding down your left mouse button on the arrow and move your mouse up and down to resize the chair.



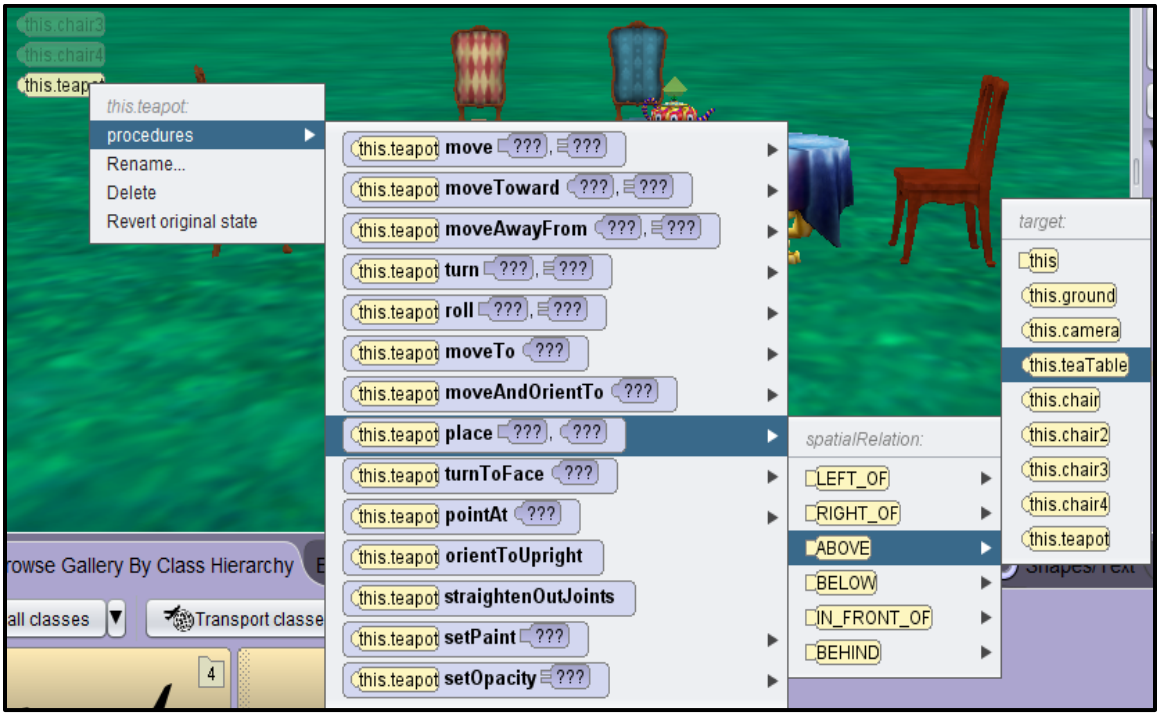
13. To rotate the chair, click on the **rotation** button from the handle style choices. If you hold down the left mouse button on the bottom ring and drag to the right and left, it will spin the chair around so that you can have it face the table. To move the chair, click on the **move** button from the handle style choices. If you hold down your left mouse button on the arrow on top of the chair and drag up and down, the chair will move up and down. The arrow in the front will move the chair forward and backward. The arrow to the right will move the chair left and right.

14. **Add 3 more chairs** to the scene around the table. Be careful not to give the chairs the same name. You will see the following error if you try to name your objects the same name. You can call the other chairs: chair2, chair3, and chair4. Do not put spaces in your names. The Alice software will not allow you to name your objects with spaces and this is because the Java language does not allow you to have spaces when naming.



15. It should look similar to the following:

16. Next, we need to add some teacups and a teapot onto the table. If you search for tea in the gallery, you will be given the teapot, teacups, saucers, etc. I would like to start with the teapot. When you create the teapot, you can use the default name. We can play with trying to get this teapot onto the table, but this would take a while and there is an easier way. If you right click on the teapot, select procedures, teapot place…, above, and teaTable, it will place the teapot on top of the table for you.

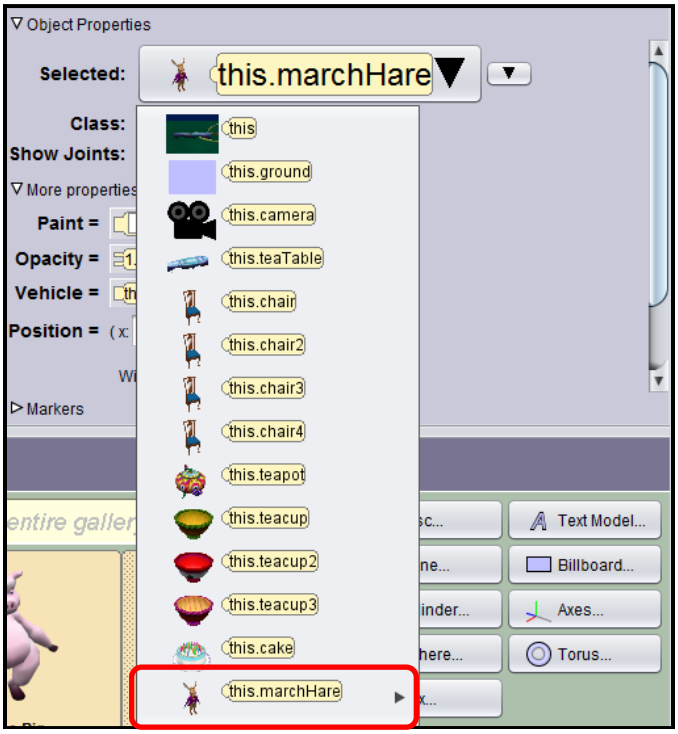


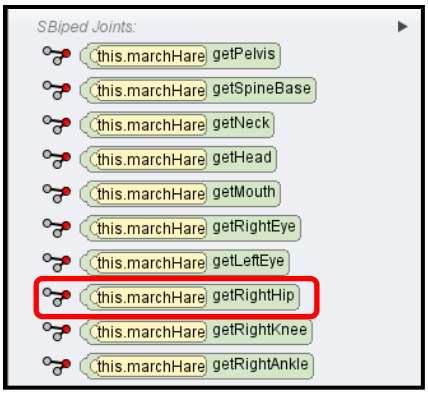
17. **Add a few teacups onto the table** and adjust them how you want them. *Be careful not to give 2 teacups the same name.*

18. **Add a birthday cake onto the table** and readjust the items on the table. It should look similar to the following.

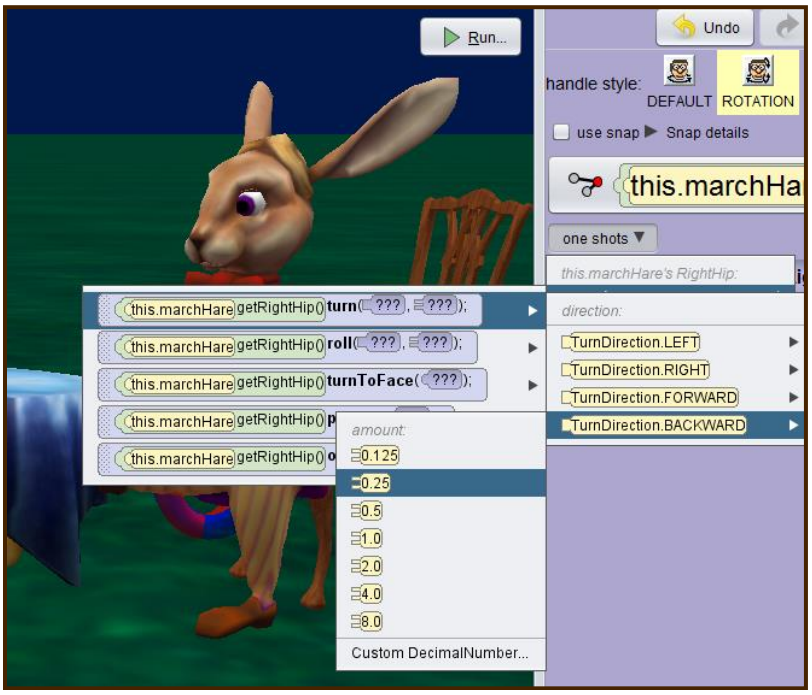
19. Next, we are going to add the characters. The characters can be found in the biped folder in the gallery. Let's add the March Hare first. Place him directly in front of one of the chairs. It doesn't matter which chair you choose. You will need to rotate him so that he lined up with the chair. We are going to make him sit in the chair.

20. To move the marchHare's joints, we will need to select the marchHare and drop down his subparts as shown below. Choose the hare's right hip.





Now, we need to select **ONE SHOT, procedures**, **marchHare.getRightHip.turn…**, **BACKWARD**, and **0.25**



21. **Repeat this for the leftHip**.

22. Select the **marchHare's rightKnee**, then select **one shots**, **procedures**, **turn**, **forward**, and **0.25**.

23. **Repeat this for the leftKnee**. You may need to move the entire marchHare back and up to get him onto the chair.

24. Now, let's **add the madHatter** to the scene. Place him next to the marchHare. It doesn't matter which side he is on. You may need to resize, rotate, and move him to get the scene to look the way you want.

25. Finally, we are going to add Alice to the scene. We will need to create Alice using the Child class in the biped classes. The Child class allows you to select male or female, the skin tone, the attire, the hair color, eye color, and shape of the person. **Create a girl** that looks like Alice and name her **alice**. Normally you would capitalize a name, but when we name objects, we don't capitalize the object names.

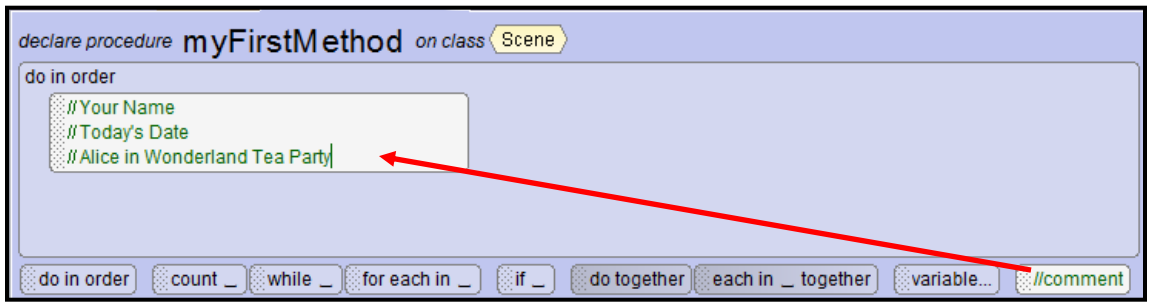
26. Place alice off to the side of the animation window looking at the tea party as shown below.

27. We are finished with the scene setup. If you want to add some wonderland trees or other objects to your scene, feel free.

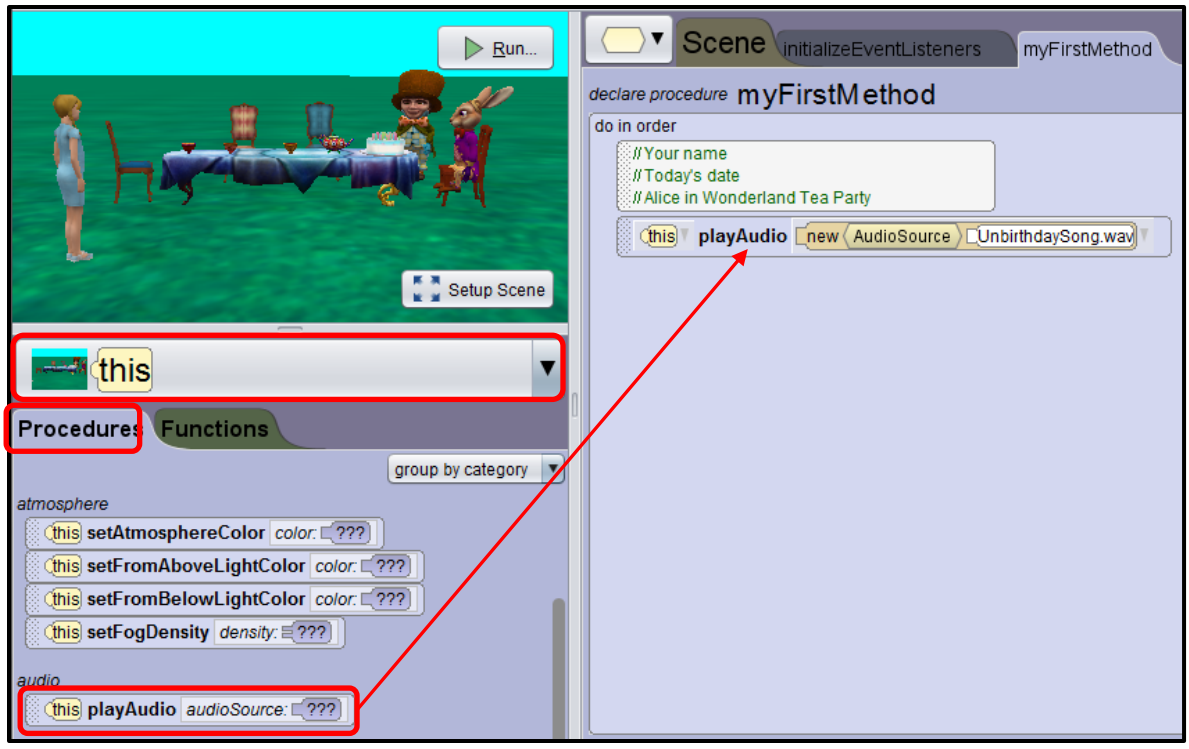
28. Before writing the code for our animation, we should first create a storyboard of what we wish to accomplish.

- Scene opens with the Mad Hatter and the March Hare gathered around a table with tea and a birthday cake
- The unbirthday song plays
- Alice approaches the table
- Alice tells the characters that she enjoyed their singing
- They tell her that nobody ever compliments their singing and insist that she has a cup of tea
- She apologizes for interrupting their birthday party
- They explain that it isn't their birthday; it is their "unbirthday"
- Alice then asks them to explain an "unbirthday"
- They then tell her that everyone has 364 "unbirthdays" each year
- Alice realizes that it is her "unbirthday" too

29. Drag the **//comment** block to the editor and enter your comments. You need to put your name, the date, and a description of the program in comments at the top of all of your programs.

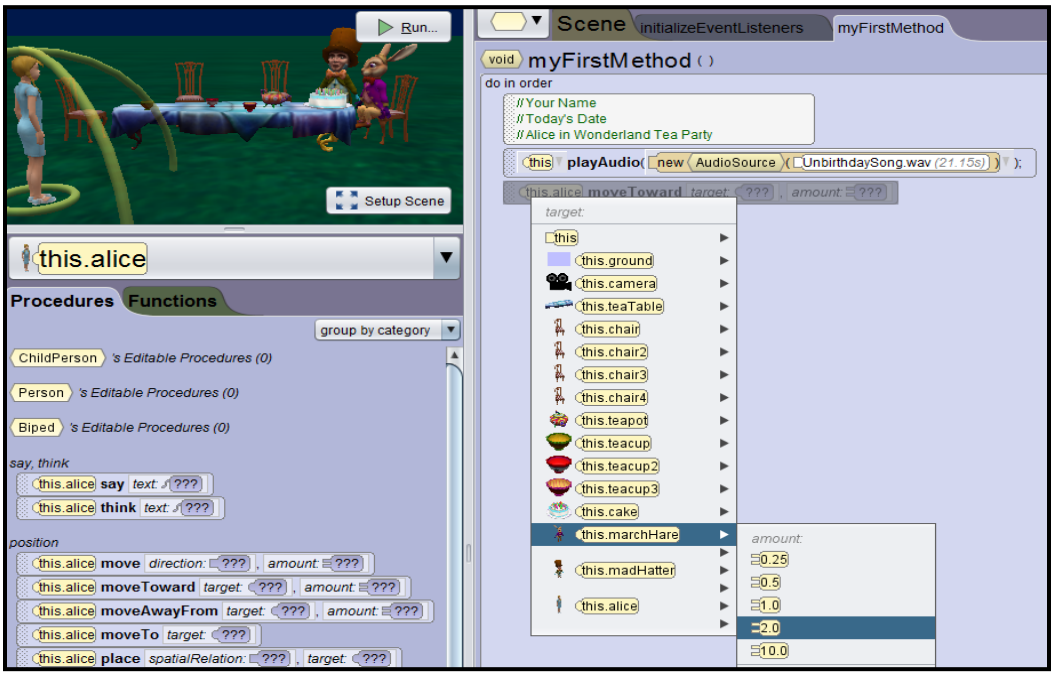


30. Click on the **this**, make sure that the **Procedures** tab is selected, drag the **playAudio** method to the editor, select **Import Audio**, use the **Unbirthday Song** file *(located in your Data_Files folder:* http://faculty.collin.edu/tdaly/book4/*)*. Procedural methods are actions that objects can do. The word "this" refers to the scene and we are telling the scene to play the audio.

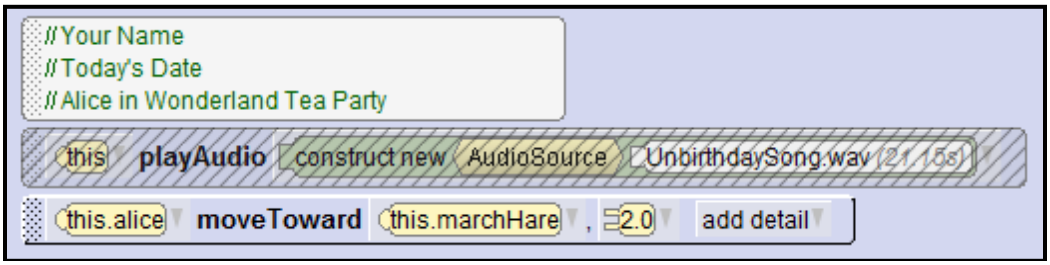


31. We need to test the program by clicking the **Run** button to play the animation. You should hear the song play but nothing else happens yet.

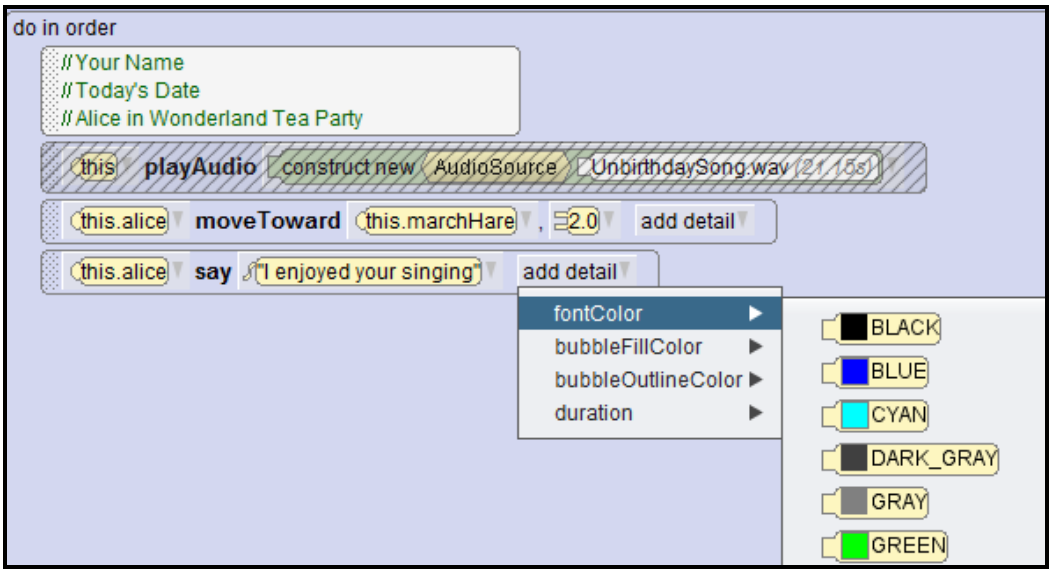
32. Next, we will select alice, make sure that the procedures tab is selected, and drag the **moveToward method** onto the editor underneath the playAudio method. When you release the mouse, you will be prompted to select the target that you want alice to move toward and the amount that you want her to move. Select **marchHare** as the target and **2.0** as the amount. If you wanted a number that isn't on the list, you would select Custom DecimalNumber and type in your own number. These choices (target and amount) are known as arguments in programming. Run the animation to see if alice moves toward the marchHare; you will have to wait until the song finishes to see her move. The program happens in order. The next line doesn't execute until the previous line is finished.



33. If you don't want to wait for the song to finish every time you want to test out your animation, you can disable lines of code and enable them later. To do this, you would need to right click on the **playAudio** line and **uncheck Is Enabled**. You will see the line will now have gray lines over it.



34. Now, we want alice to praise their singing. You will need to select alice and then drag the **say** method onto the editor underneath the moveToward method. When you do this, you will be prompted (argument) to enter the text of what you want alice to say. You should select **Custom TextString…** and then enter **I enjoyed your singing**. You can select **add detail** if you want to make adjustments such as text color, speech bubble color, outline color, or the duration that the bubble stays on the screen. You can leave the default settings if you want. The duration is defaulted to 1 second.

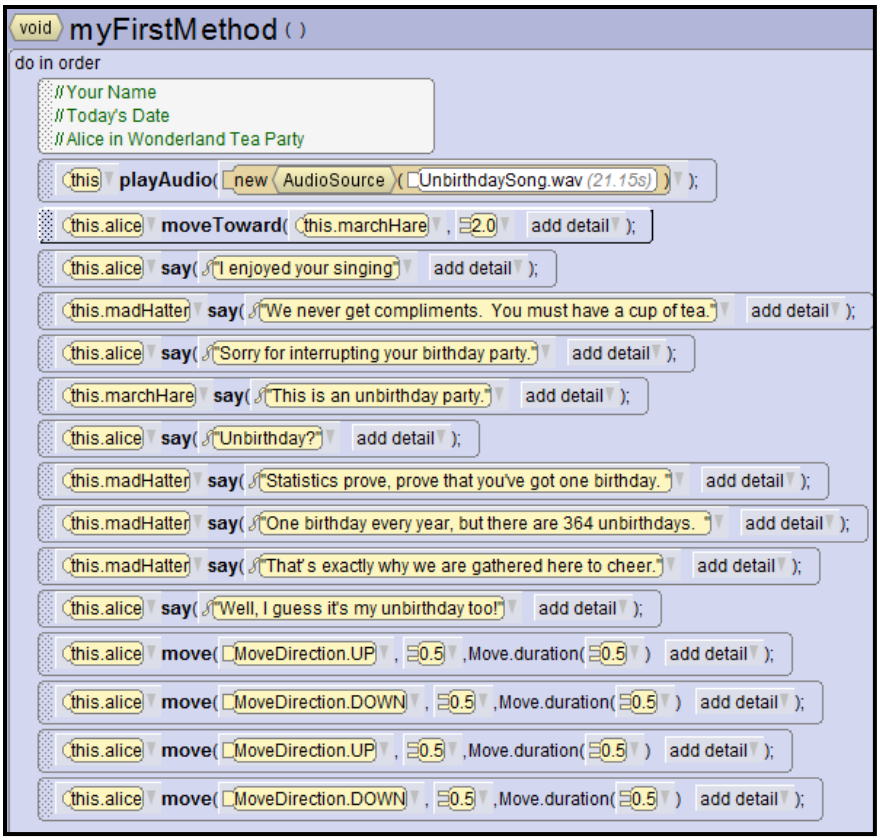


35. Create the following dialog between the characters:

madHatter – We never get compliments, you must have a cup of tea.
alice – Sorry for interrupting your birthday party.
marchHare – This is an unbirthday party.
alice – Unbirthday?
madHatter - Statistics prove, prove that you've got one birthday. One birthday every year, but there are 364 unbirthdays. That's exactly why we are gathered here to cheer.
alice – Well I guess it's my unbirthday too!

36. Have Alice joyously jump up and down at the end. If you want her to jump at a faster pace, you can change the duration to be 0.5 seconds instead of 1 second (Click the *add detail* drop down to change the duration.

37. To test the full program with the song, you will need to **enable** the **playAudio** method. Right click on the **playAudio** line and **check Is Enabled**. The grey lines through the playAudio method should disappear.



38. Save your work and exit Alice.

Summary

- Alice is an innovative 3D programming environment that makes it easy to create an animation or interactive game. The team named the system "Alice" in honor of Lewis Carroll who wrote Alice's Adventures in Wonderland.
- An Alice virtual world begins with a template for an initial scene. These templates can be grass, water, snow, etc.
- An Alice 3D model is like a blueprint that tells Alice how to create a new object in the scene. The 3D model provides instructions on how to draw the object, what color it should be, what parts it should have, its size (height, width, and depth), and many other details.
- When you choose to place an Alice object into your world, Alice will create an object (instance) of that class in your world and ask you to name that object.
- When naming an object (instance) of a class, you should begin the name with a lowercase letter. If the name will have multiple words in it, each successive word with then begin with a capital letter. An example would be ***myLittleSnowman***.
- Objects from the galleries are added to the scene via the SCENE EDITOR (a click on Setup Scene button).
- Objects in an Alice world are three dimensional. Each object has width, height, and depth.
- There are six possible directions in which an object may move – forward, backward, up, down, left and right.
- Each object in Alice has a unique "center." An object's center is used for measuring distance to another object and for determining its position in the world.

Review Questions

1. Alice was named in honor of Lewis Carroll.
    a. True
    b. False

2. An Alice 3D model is like a blueprint that tells Alice how to create a new object in the scene.
    a. True
    b. False

3. Once an object is placed into a scene, it can't be manipulated by moving, rotating, etc.
    a. True
    b. False

4. If you were to name an object (instance) of an Airplane class, which of the following names would be proper?
    a. MyAirplane
    b. my airplane
    c. myairplane
    d. myAirplane

5. You can have more than one object of the same class in the same world?
    a. True
    b. False

Solutions: *1) a 2) a 3) b 4) d 5) a*

Assignments

0-1 Cola Commercial: Your goal is to create a cola commercial using Alice and NetBeans.

- ***Analyze and understand the problem to be solved.*** We would like to create a cola commercial animation that is at least 5 seconds long. We need to take a look at the gallery to see what objects we have that could be used in our commercial. Then we need to set up the scene.
- ***Develop the logic to solve the program.*** We should develop a storyboard for our animation. The storyboard is a short description of what you want to happen in your animation.
- ***Code the solution in a programming language.*** Write the code in Alice. Give the file an appropriate name. Add your name, date, and a description of the program to the top of the program as comments.
- ***Test the program.*** Test the code in Alice.

0-2 Greeting Card: Your goal is to create an animated greeting card using Alice and NetBeans.

- ***Analyze and understand the problem to be solved.*** We would like to create a greeting card animation that is at least 5 seconds long. We need to take a look at the gallery to see what objects we have that could be used in our commercial. Then we need to set up the scene.
- ***Develop the logic to solve the program.*** We should develop a storyboard for our animation. The storyboard is a short description of what you want to happen in your animation.
- ***Code the solution in a programming language.*** Write the code in Alice. Give the file an appropriate name. Add your name, date, and a description of the program to the top of the program as comments.
- ***Test the program.*** Test the code in Alice.

0-3 Animation: Your goal is to create a short animation using Alice and NetBeans.

- ***Analyze and understand the problem to be solved.*** We would like to create a short animation of our choosing that is at least 5 seconds long. We need to take a look at the gallery to see what objects we have that could be used in our commercial. Then we need to set up the scene.
- ***Develop the logic to solve the program.*** We should develop a storyboard for our animation. The storyboard is a short description of what you want to happen in your animation.
- ***Code the solution in a programming language.*** Write the code in Alice. Give the file an appropriate name. Add your name, date, and a description of the program to the top of the program as comments.
- ***Test the program.*** Test the code in Alice.

Chapter 1

Coding Introduction

Objectives

- ☑ Explain the difference between high and low level programming languages
- ☑ Describe the history of how the Java programming language was started
- ☑ Briefly describe the following:
  - Object Oriented Programming
  - Platform-Independence
  - Garbage Collection
  - Java Development Kit
- ☑ Explain the difference between applets, applications, and servlets
- ☑ Explain the difference between Java and JavaScript
- ☑ Compile and execute a Java program
- ☑ Debug errors
- ☑ Identify and fix compiler errors

Introduction to Programming

A computer program is a way to tell a computer what to do. When you want a computer to perform a task, you must give it line-by-line instructions on how to accomplish that task. These line-by-line instructions are called a **computer program**.

The computer stores information based on electronic signals, referred to as **binary**. A **bit** (binary digit**)**, the smallest unit of information storage, is represented by either an on (1) or off (0) signal inside the computer. One **byte** (a character such as the letter "A" on the keyboard) uses eight bits.

There are many different computer programming languages available and the choice of what programming language to use will depend upon the task for the computer to accomplish. A programming language that is written at the very low technical circuitry level of the computer is called a **low-level programming language**. Some examples of low-level programming languages are machine language and assembler language. Machine language is composed of binary 1's and 0's and is not intended for humans to read. Machine language varies from computer to computer. The machine language for a PC is entirely different from machine language for Mac. A computer only understands programs (without any conversion) written in its machine language (binary).

High-level programming languages allow programmers to write programs using English terms. Computers do not understand high-level languages directly so this means that computer programs written in a high-level language must be converted to machine language by an interpreter or compiler. Some high-level computer programming languages available are: C++, Visual Basic.NET, C#, and Java. Each of these programming languages is best-suited to a certain type of computer or problem such as mainframes, business, games and/or science.

Computer languages each have their own **syntax**, or rules of the language. For instance, in a high-level programming language the verb to display information might be "write", "print", "show", etc. In a low-level programming language the verb to display information might be a code of "101011" in binary. ***Java is a high-level programming language with a specific vocabulary and specific rules for using that vocabulary.***

What is Java?

History of Java

In 1990, **James Gosling** was given the task of creating programs to control consumer electronics (TVs, VCRs, toasters, etc.). Gosling and his team at **Sun Microsystems** started designing their software using C++. The team found that C++ was not suitable for the projects they had in mind. They ran into trouble with complicated aspects of C++ such as multiple inheritances of classes and with program bugs such as memory leaks. So, Gosling created a simplified computer

language that would avoid all the problems he had with C++. Thus, **a new programming language named Oak** (after a tree outside his window) was born.

Oak was first used in something called the Green project, which was a control system for use in the home using a hand-held computer called Star Seven. Oak was then used in another project which involved video-on-demand. Neither project ever made it to the public eye, but Oak gained some recognition. Sun discovered that the name Oak was already copyrighted. After going out for coffee one day, they named their new powerful language **Java**.

In 1993, the Java team realized that the Java language they had developed would be perfect for web page programming. The team came up with the concept of web applets, small programs that could be included in web pages, and created a complete web browser called HotJava (originally called Webrunner) that demonstrated the language's power.

In the second quarter of 1995, Sun Microsystems officially announced Java. The "new" language was quickly embraced as a powerful tool for developing Internet applications. Netscape Communications added support for Java to its Netscape Navigator 2.0. Java became an instant "hit" and also made the Netscape browser very popular. Other Internet software developers such as Microsoft eventually followed suit and reluctantly included Java in their browsers. These browsers were called "Java-enabled". Java-enabled meant that the browser could download and play Java classes (applets) on the user's system. (Applets appear in a web page much the same way as images do, but unlike images, applets can be dynamic and interactive.)

Java Capabilities

- **Java is easier than C++**. Although Java looks similar to C and C++, most of the complex parts such as pointers, multiple inheritance, and memory management have been excluded from Java.

- **Java is an Object Oriented Programming (OOP) language,** which allows you to create flexible, modular programs and reuse code. OOP is based on the theory that everything in the world can be modeled as an object. An object has attributes (data) and behavior (methods).

- **Java is platform-independent.** Platform-independence is a program's capability of moving easily from one computer system to another. **Java's slogan is "You can write once and run anywhere."** If you write a game using the Java programming language, theoretically, you should be able to run that game on a PC, Linux, or Mac.

- **Java supports the Internet** by enabling people to write interactive programs for the Internet. Java applets can easily be invoked from web browsers to provide valuable and spectacular web pages.

- **Java is general purpose.** Although used mainly for writing internet applications, Java is a truly general-purpose language. Almost anything that most other computer programming languages such as C++ or Visual Basic can do, Java can also do. Java programs can be applets for the Internet or standalone applications for local PCs.

    - **Applets** appear in a web page much in the same way as images do, but unlike images, applets are dynamic and interactive. Applets can be used to create animations, games, ecommerce, etc.

    - **Applications** are more general programs written in the Java language. Applications don't need a browser. The Java language can be used to create programs, like those made in other computer languages.

    - **Servlets** are programs that respond to requests from clients.

- **Java is secure**. Since the Java program is isolated from the native operating system of a computer, the Java program is insulated from the particular hardware on which it is run. Because of this insulation, the Java Virtual Machine provides security against intruders getting at your computer's hardware through the operating system.

- **Java programs can contain multiple threads** of execution, which enables programs to handle several tasks simultaneously. For example, a multi-threaded program can render an animation on the screen in one thread while continuing to accept keyboard input from the user in the main thread. All applications have at least one thread.

- **Java has multimedia capabilities** of graphics, images, animations, audio and videos. It also runs on networks.

- **Java programs do their own garbage collection**, which means that programs are not required to delete objects that they allocate to memory. This relieves programmers of virtually all memory-management problems.

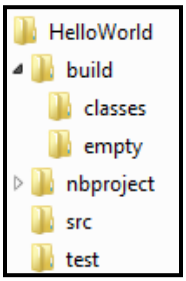
- **Java programs are reliable and robust**. When a serious error is discovered, Java programs create an **exception**. This exception can be captured and managed by the program and then terminated gracefully.

- **Java vs. JavaScript**. The Java language was developed by Sun MicroSystems and is a full programming language that can be used in applications or as applets on the Internet. JavaScript was developed by Netscape as a scripting language to be used only in HTML web pages.

Programming Process

Develop an algorithm: Think about the problem before coding. Create a flowchart, storyboard, or pseudo code to represent a solution to the problem.

Create Project: Create a new project in NetBeans. The NetBeans environment is known as our **IDE** (Integrated Development Environment). There are many IDEs that can be downloaded free of charge, but NetBeans provides many features that will be helpful to us for this course. In NetBeans, when you create a project, it creates a folder structure. The following is an example of a folder structure for a HelloWorld project created in NetBeans.



Code: Type the Java code. As you type, NetBeans checks your program for syntax errors. Red lines indicate errors in your code. The Java code (HelloWorld.java) for the HelloWorld project will be in the **src** folder in the folder structure above.

Compile: When you are finished typing the program, you will need to do a final compile of the program (also known as building). The Java compiler checks your code for errors. If it compiles with no syntax errors, it creates a **class file** (bytecode) that will be capable of running on different operating systems. Bytecode are a set of instructions that look a lot like machine code, but are not specific to any one processor. Compiling the HelloWorld.java creates the HelloWorld.class file located in the **build** folder inside the **classes** subfolder in the folder structure above.

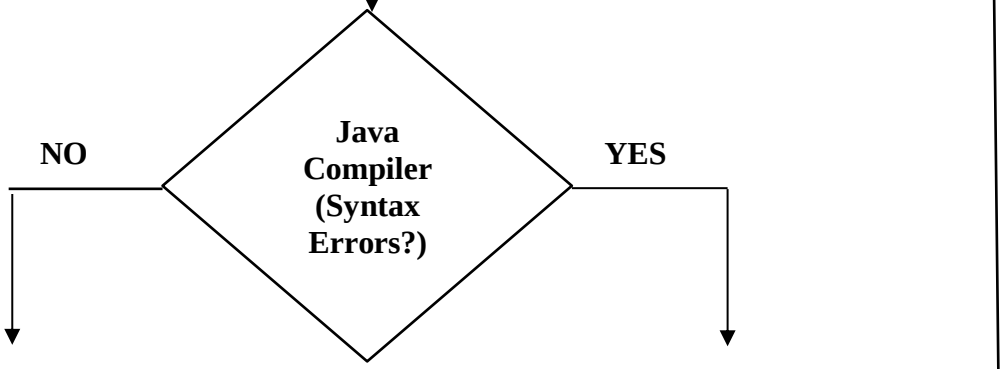
Run: These bytecode are then fed to a **JVM** (Java Virtual Machine) where they are interpreted and executed.

The **JDK** (Java Development Kit) includes the Java library (code), JVM, as well as the Java compiler. The version of NetBeans that you installed included the JDK. Oracle owns Java and it is constantly releasing new versions of the JDK. It is good to know what JDK you are using so that you know what Java code is available to you. We will talk more about the Java library and JDK in a later chapter. You can check to see what version of the JDK that you have by clicking **Help** from the menu and then **About** in the NetBeans environment. The JDK version shown below is 1.8. You do not need to put the update number which is the number after the underscore.

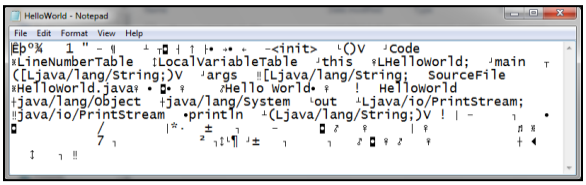
Product Version: NetBeans IDE 8.0 (Build 201403101706)
Updates: NetBeans IDE is updated to version NetBeans 8.0 Patch 1.1
Java: 1.8.0_05; Java HotSpot(TM) Client VM 25.5-b02
Runtime: Java(TM) SE Runtime Environment 1.8.0_05-b13
System: Windows 7 version 6.1 running on x86; Cp1252; en_US (nb)
User directory: C:\Users\Administrator\AppData\Roaming\NetBeans\8.0

Java Program – HelloWorld.java

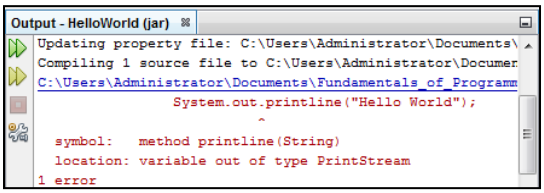
```
public class HelloWorld
{
        public static void main (String [ ] args) {
                System.out.println("Hello World");
        }
}
```



Java Class File – HelloWorld.class

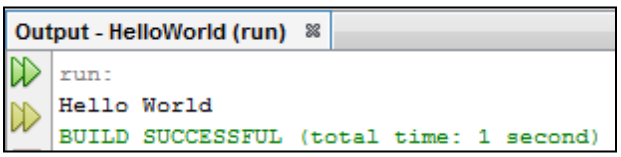


Errors Displayed in Output Window



Java Virtual Machine – Program Execution

Fix program errors



Documentation

Comments are used to document code so that other people reading our code can understand our logic. Comments are useful for adding extra information to our programs that we don't necessarily want to show up in the output of our program such as: author, date, JDK used, program description, etc. Also, it is a good idea to comment your programs extensively when you are just starting out so that you have well-documented examples.

A **single line comment** is represented by two forward slashes. This comment will continue until the end of the line. This type of comment can be placed on a line by itself or it can be placed on the end of a line of code to describe the code. The following are examples of a single line comments.

//Single Line Comment

System.out.println("Hello World"); **//Prints "Hello World" to the output window**

A **multi-line comment** is represented by a forward slash followed by an asterisk and an asterisk followed by a forward slash to end the multi-line comment. The following is an example of a multi-line comment.

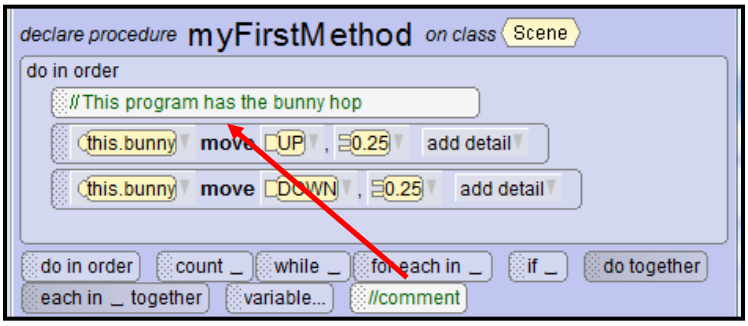
```
/* Multi-Line
   Comment */
```

You can even be creative and separate your multi-line comments from your code by adding asterisks after the first forward slash and before the last forward slash.

```
/**
 * Multi-Line Comment
 * Typical Java Documentation
 */
```

Note: Java documentation will be explained in a later chapter.

Alice Comments

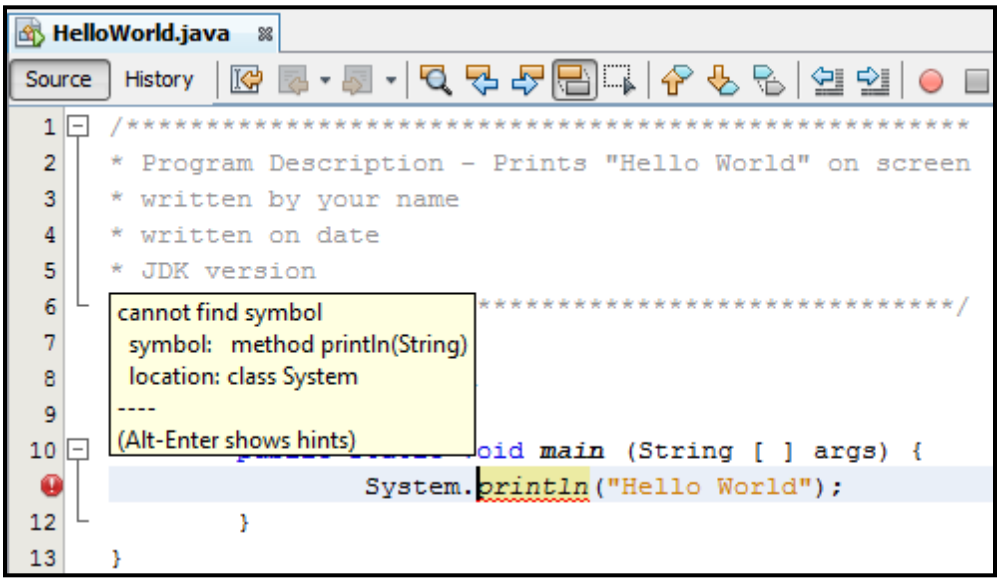


Program Errors

There are 3 different types of programming errors: compiler, run-time, and logic errors.

- **Syntax errors** are caused when the user writes code that is not understood by the compiler. A syntax error can be caused by incorrect capitalization or spelling mistakes. The compiler informs the user of a syntax error by displaying an error message. Typing "Public Class" instead of "public class" would result in a syntax error. NetBeans checks for errors as you type. If you see a red exclamation point before a line of code, you can hover over it with your mouse to see the error.

  This line of code should have been: **System.out.println("Hello World");**



- **Run-time errors** are caused by invalid data. Run-time errors do not affect the compilation of your program thus the program will compile and execute, but it may crash or hang after execution. If you try to divide 12 by 0 you would get a run-time error because you cannot divide by 0.

- **Logic errors** (also known as human error) are caused by mistakes that do not defy the rules of the language and do not crash or hang the program, but instead yield incorrect results. The user may not understand the problem that the program is trying to solve and therefore uses the wrong equation, wrong strategy, etc. An example of a logic error would be moving left instead of right.

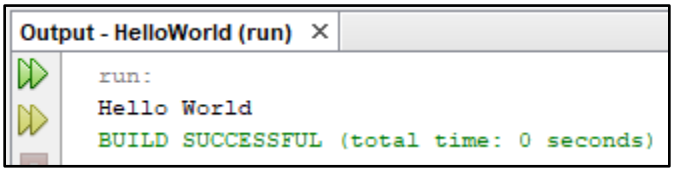
Java Basics

Statements

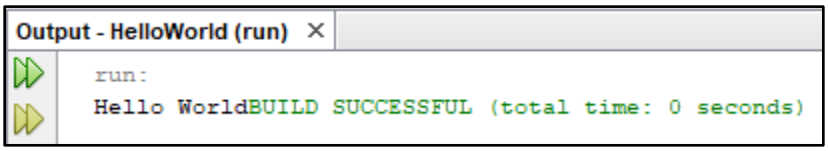
A statement is the simplest thing you can do in Java. A Java statement forms a single Java operation. Each statement generally ends with a semicolon. The following statement will print "Hello World" to the screen. The **println** command, known as a method in Java will do the printing and the word **out** signifies the object to to the printing to which is the screen. The "Hello World" is the text to be printed, the text in Java is known as a string. Java string literals are written as a sequence of characters in double quotes.

System.out.println("Hello World");

This **println** method will print the following to the output window.



You can also use the **print** method in Java, but your text will not appear on a new line in the output window. The **println** will cause the "Hello World" to be printing on separate line. The **ln** stands for line.



We will also be using coding that will be known as blocks of code. Blocks of code are surrounded by curly braces { }. We will be putting our statements that we write into a block of code known as the **main** method. When we run our programs to see the results, everything in the **main** method will happen automatically. You will not understand the words that are used to create this method until later in the text, but just know for now that every application that we create will have this **main** method and it will automatically be created for you and it will automatically run when you run your program.

```
public static void main(String[] args) {
    System.out.print("Hello World");
}
```

The **main** method will be located inside of our **class**. Our class represents our file and the main method should always be inside of it. You will not understand the words used to create the class file at this time, but we will talk about this later in the text.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.print("Hello World");
    }
}
```

We could keep writing **println** statements in order to print multiple lines of code. Be sure to use **println** instead of **print** to get separate lines. We will practice this in the hands-on exercises.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
        System.out.println("We are learning how to code.");
    }
}
```

```
Output - HelloWorld (run)
run:
Hello World
We are learning how to code.
BUILD SUCCESSFUL (total time: 0 seconds)
```

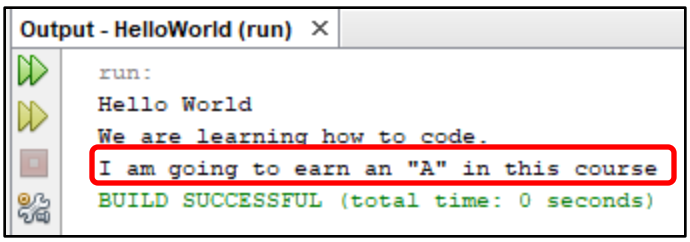
Escape Codes

You can use escape codes to add new lines, quotes, etc. to your strings.

\n	Newline
\t	Tab
\\	Displays a Backslash
\'	Displays a single quote
\"	Displays a double quote

Strings can contain character escape codes such as the " (double quote) or ' (single quote) by including a backslash in front of it. The \" tells it to print the special character of a double quotes.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
        System.out.println("We are learning how to code.");
        System.out.println("I am going to earn an \"A\" in this course");
    }
}
```

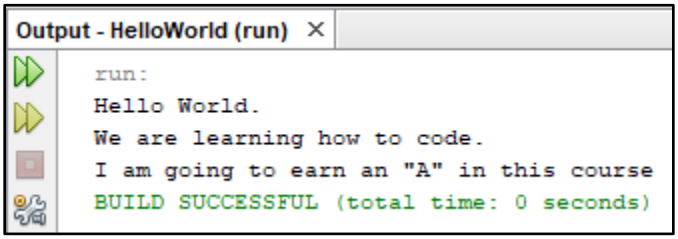


We could use the **\n** escape code to get new lines in our code instead of using a separate **println** statement for every line.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World. \nWe are learning how to code. \nI am going to earn an \"A\" in this course");
    }
}
```

You will notice that the code is past the red line in NetBeans. This red line shows the printing area. Anything past the red line will be printed on a new sheet if you were to print your code. We should try not to write our code past this line. If you hit enter before the **\n** after the following sentence: "We are learning how to code." You will notice that NetBeans creates two strings joined by a plus sign. This is known as **string concatenation**. You will notice that joining the string on separate lines, does not change the output. You need to add your **\n** escape codes inside your strings to add new lines to your output result. Be careful not to add extra spaces before and after your escape codes or Java will add extra spaces to your actual output. We will practice this in the hands-on exercises.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World. \nWe are learning how to code. "
                + "\nI am going to earn an \"A\" in this course");
    }
}
```



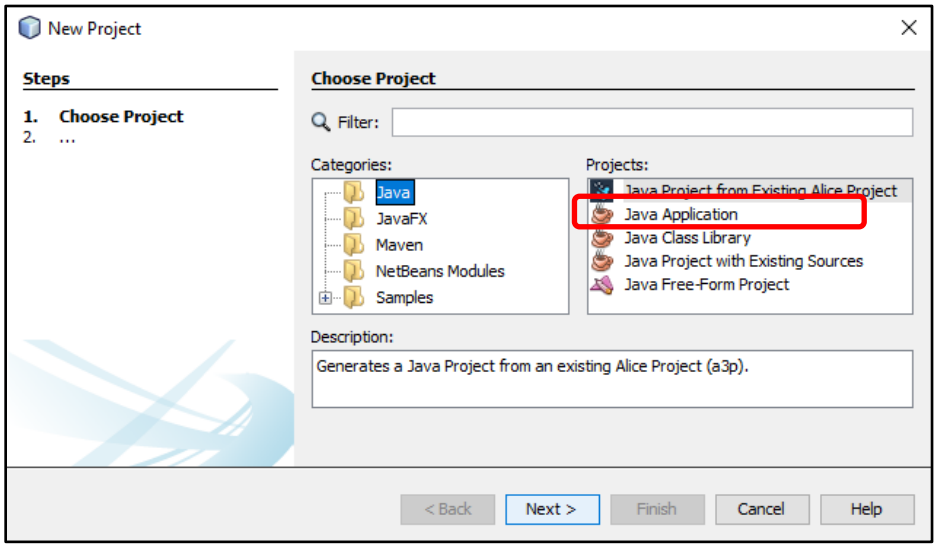
Hands-on Exercises

Exercise 1: Compiling and Executing a Java Program

1. Open up the **NetBeans** environment.

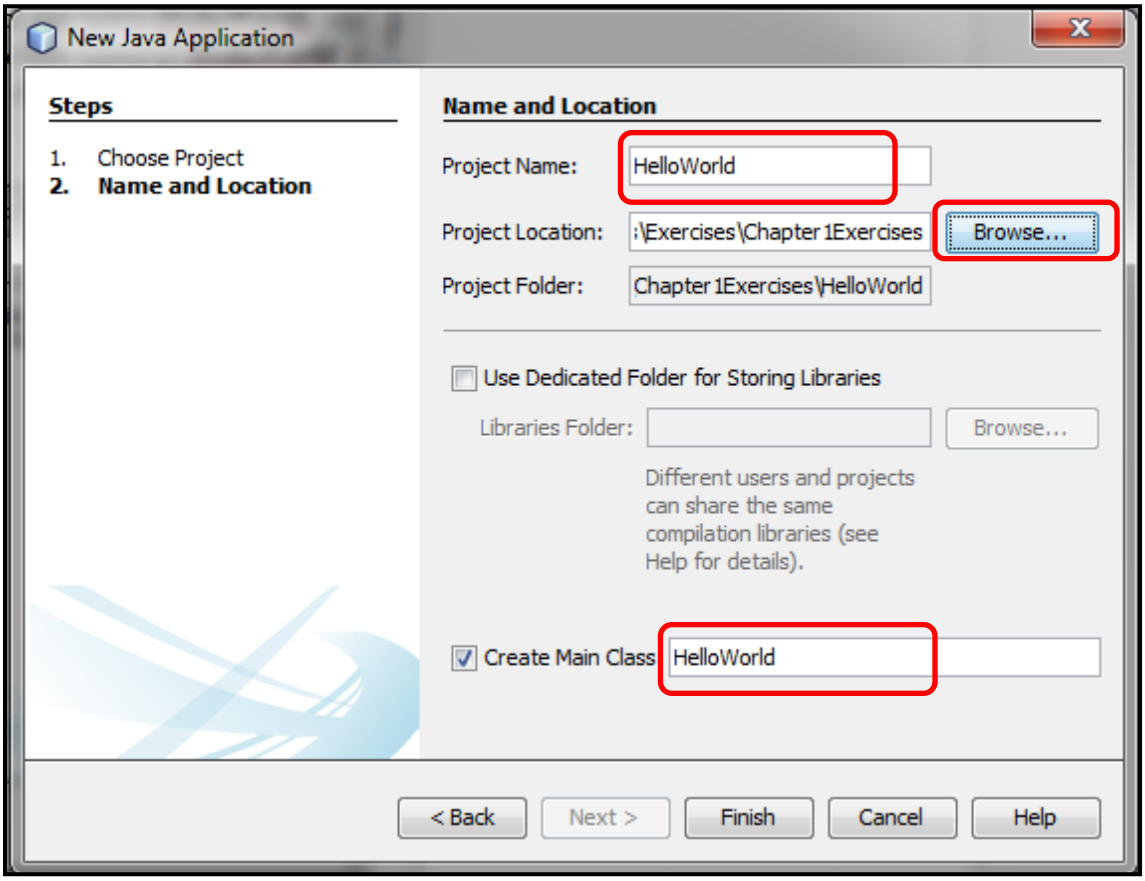
2. You can close the **Start Page**. The tutorials provided in the Start Page can be confusing for a first timer.

3. Select the **File** menu and then choose **New Project**. Then choose **Java Application** as shown below.



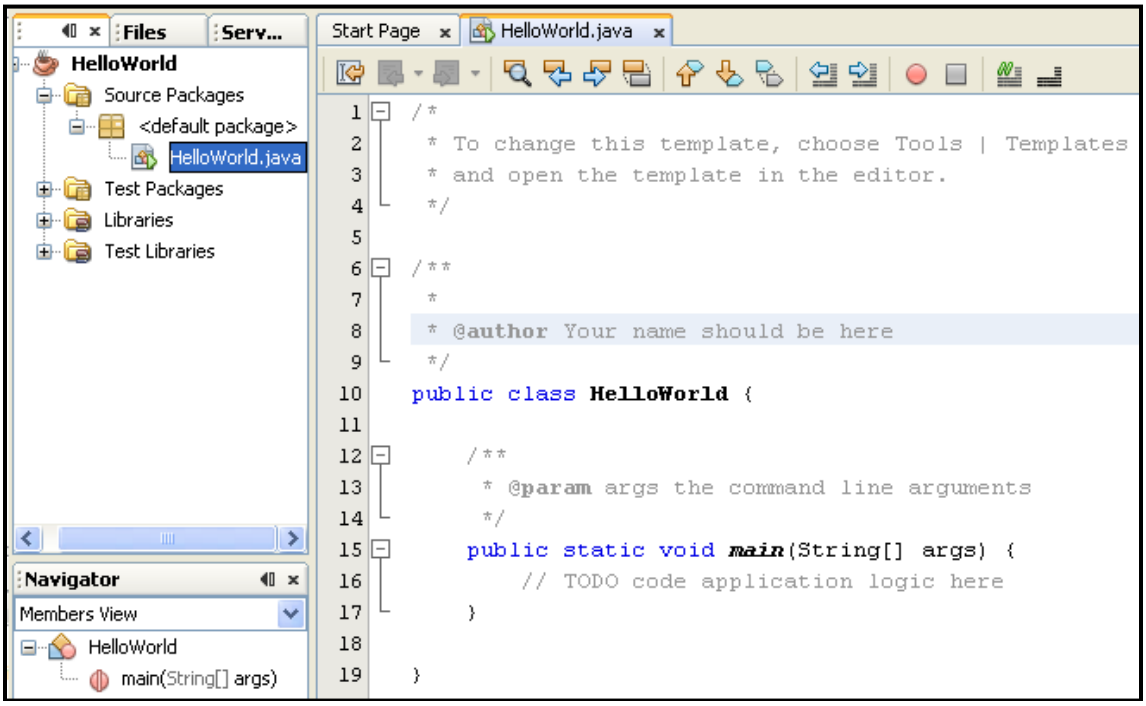
4. Click **Next**. Name your NetBeans project, select the **location** of where you would like to save your file, give your file *(Main Class)* a name *(make this name the same as your project name)*, and click **finish**. Although it is not necessary, we are going to name our projects and Java files *(main class)* have the same name. Therefore, make sure that the top and bottom boxes have the same name. NetBeans automatically will try to name your file (main class) helloworld.HelloWorld. Erase the **helloworld.** that NetBeans inserts before your file name. Make sure it looks like the following screenshot. Capitalization is important.

    *Project the name **HelloWorld** (no spaces)*
    *Main class name of **HelloWorld** (no spaces)*
    *Select the **Location** of where you would like to save your NetBeans project*

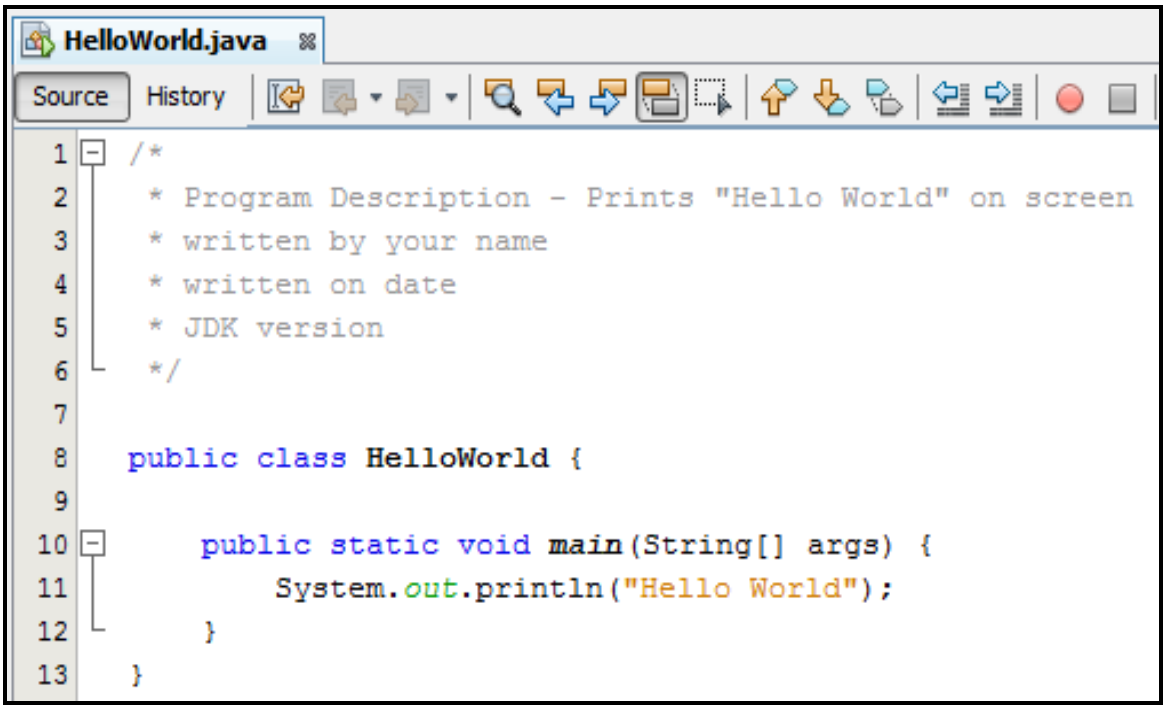


5. If line numbers are not showing, click **View** from the menu, then **Show Line Numbers.**

6. Your project should look as follows. HelloWorld.java is the file that we will be working with. *(Note: If your code has a package statement on line 5, then exit NetBeans and delete the project folder and do step 4 again. Make sure it looks like the screen shot provided. Look carefully at the textbox next to the Create Main Class label in the New Java Application dialog. For your information a package in Java is a folder. The package line indicates that you put your file in a folder when you created the project.)*



7. Type in the following Java program. You will need to delete some comment lines and add some lines. Be careful to make your program look exactly as shown below. Try to keep your statements on the same lines as those shown below and also try to use the same approximate indentation to make your program more understandable. Change line 3 to be your name, line 4 to be today's date, and line 5 to be the JDK version that you are using *(You can check to see what version of the JDK that you have by clicking **Help** from the menu and then **About** in the NetBeans environment. It will have **Java:** and then a number, this is your JDK version. You do not need the number after the underscore).*



What does the above program do? You will not understand everything about this program YET. However, here is a brief explanation line by line:

1-6) Lines 1-6 are known as comments. Comments are used to document code so that other people reading our code can understand our logic. Comments are useful for adding extra information to our programs that we don't necessarily want to show up in the output of our program such as: author, date, JDK used, program description, etc. Also, it is a good idea to comment your programs extensively when you are just starting out so that you have well-documented examples. This is a multi-line comment which is represented by a /* at beginning of comment and */ to end the multi-line comment.

7) Blank line for readability purposes. Does nothing. (not necessary)

8) States this will be a public program called HelloWorld. Class names should begin with a capital letter. Be careful of capitalization in Java programs.

9) Blank line for readability purposes. Does nothing. (not necessary)

10) This is the main method declaration in this Java application. Every Java application must contain a main method and it must always be public static. The arguments for a method always appear in parentheses. In this case, the argument is a String array called args. The variable name of args can be whatever the programmer wants it to be, but most programmers use the variable name of args. The square brackets appearing after the word String are found to the right of the "P" key on keyboard.

11) The statement of **System.out.println("Hello World");** prints Hello World to the screen and positions the insertion point on the next line. *System* is a Java class in the library and the *out* object is the screen. Java is case sensitive so be careful of capitalization. Java uses a punctuation method of class-dot-object-dot-method syntax. All methods have arguments in parenthesis which is a way of telling a

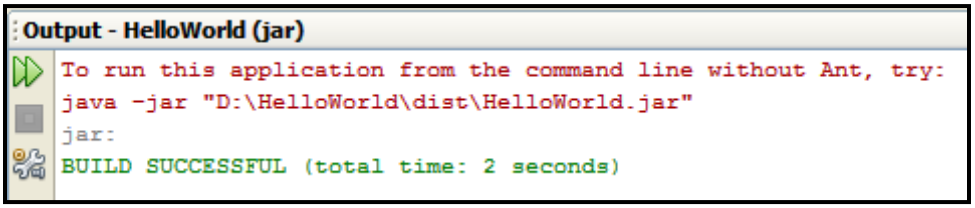
method from a variable. This println method has an argument of a literal string of "Hello World". All Java statement lines will end with a semicolon.
Note: The next to last character in "println" is a lowercase L, not the number 1.

12) A right curly brace ends the main method. It is important to balance all your left and right curly braces and left and right parentheses in all Java programs. The right curly brace is found 2 keys to the right of the P key on keyboard.

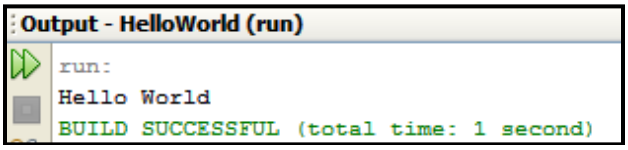
13) A right curly brace to end the program.

8. This Java program needs compiled. Compiling a program will have the computer look at each line of your program for syntax errors such as typos, mispunctuation, etc. To compile your program, click on **Run** from the File menu, then **Build Project**. The compiler will check this file for syntax errors and let you know on what lines you made errors. Errors (along with line numbers of errors) will list in bottom panel of the screen. If you have errors, correct your typos on the top of the screen and compile again. Make sure you adjust the bottom output panel large enough to see your errors and your output.

9. If there are no compilation errors (denoted by the words BUILD SUCCESSFUL), the compiler will convert this Java program into a bytecode file called ***HelloWorld.class*.** This bytecode file is a generic file that may be used on any operating system. This file is located under the **project** folder, under the **build** folder, and in the **classes** folder.



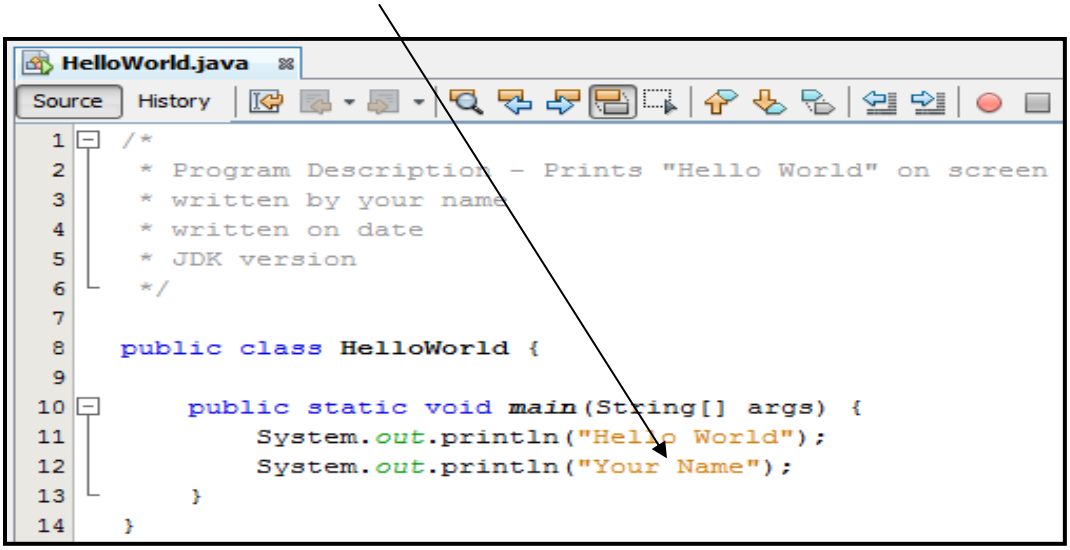
10. Once compiled and you have a bytecode file (*.class* file extension), you are ready to have the Java interpreter execute your Java program. To execute your first Java program, you will click on **Run** from the NetBeans menu, then **Run Project**. "Hello World" should be displayed in the output window as shown below:



11. The process you have seen so far is typing a Java program into NetBeans, compiling a Java program, and executing the Java program. This is the process that you will be going through over and over as you progress through Java. The output that you have at bottom of screen is simply the computer displaying the words "Hello World".

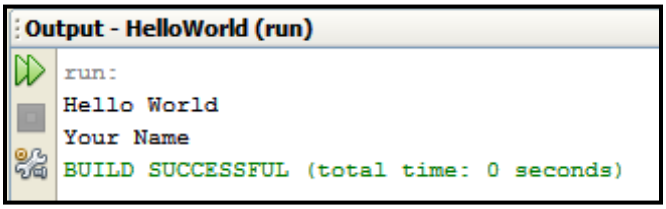
12. Now, let's adjust the Java program. Add the following line as shown in the diagram below. *(Note: if you type **sout** and hit the **tab key**, it will type the **System.out.println("");** line for you.)*

    System.out.println("Your name");



13. Now, save the new version of the program by clicking on **Save** from the File menu. (**DO NOT** click "Save As" and save this outside the project folder. NetBeans has a file structure and you cannot pull your files out of this structure or else NetBeans will not open them in the future. Compile the program (**Run** menu and then **Build Project**).

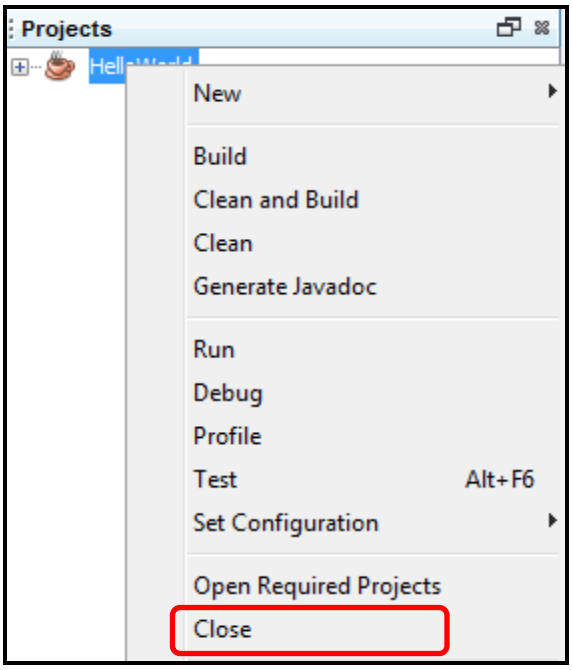
14. Execute the program (**Run** menu and then **Run Project**). Your display window should look similar to the following:



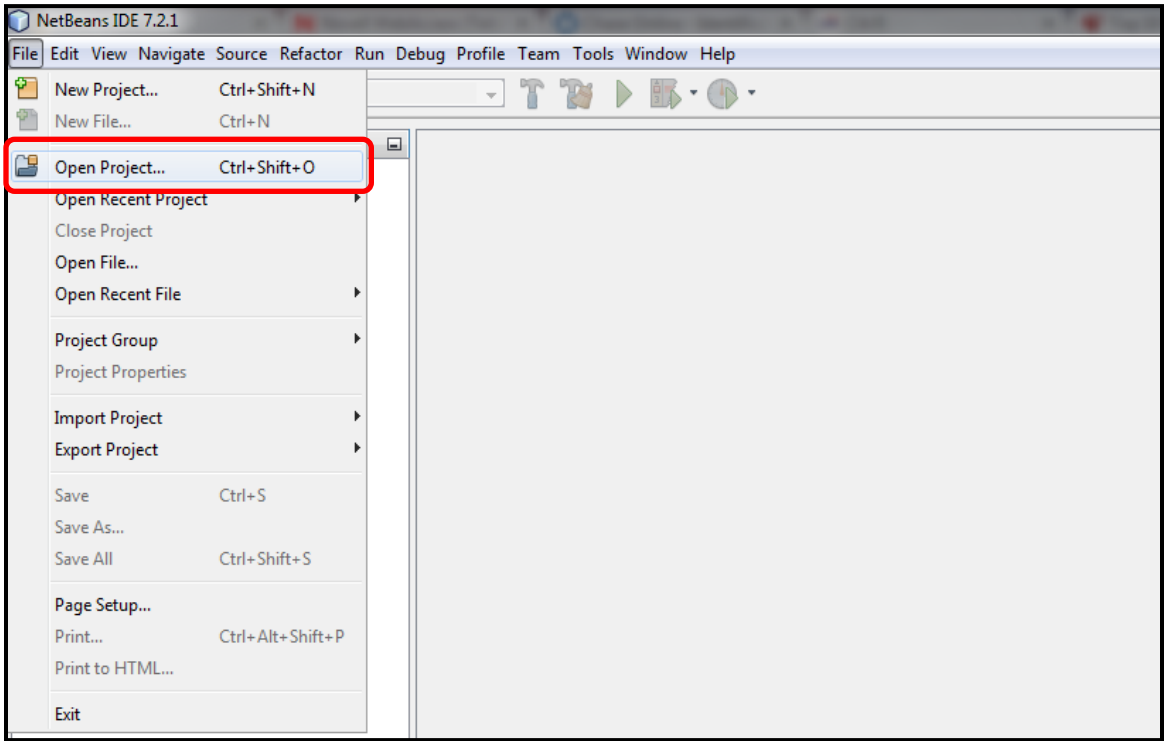
    Note: If you are getting compiler errors at bottom of screen, please double check the capitalization, spelling, and punctuation.

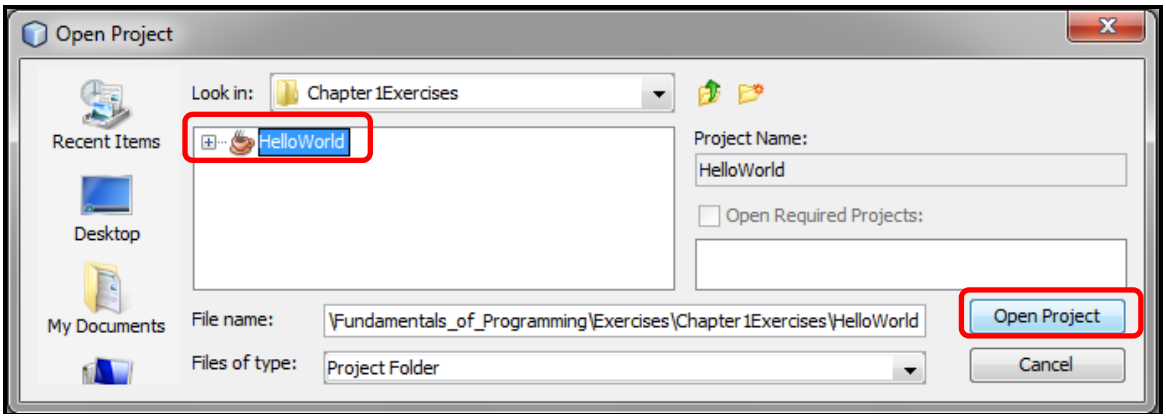
15. To ensure that your code indentation is correct, you should always choose **Source** from the menu, then **Format**. Make sure that you compile your program (Run menu, Build Project) and run it (Run menu, Run Project).

16. Close the project by right clicking on the project on the left pane and choosing **Close**.

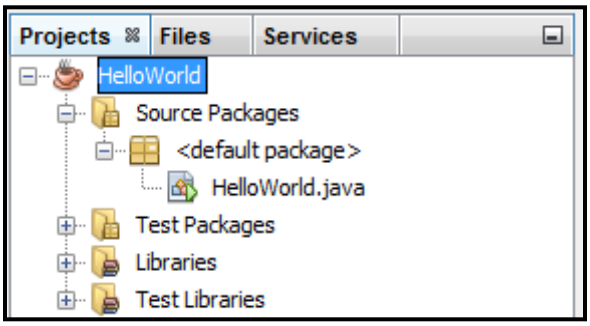


17. Choose **File**, **Open Project…**, select the **HelloWorld** NetBeans project (should have a coffee cup next to your project), and click **Open Project**.





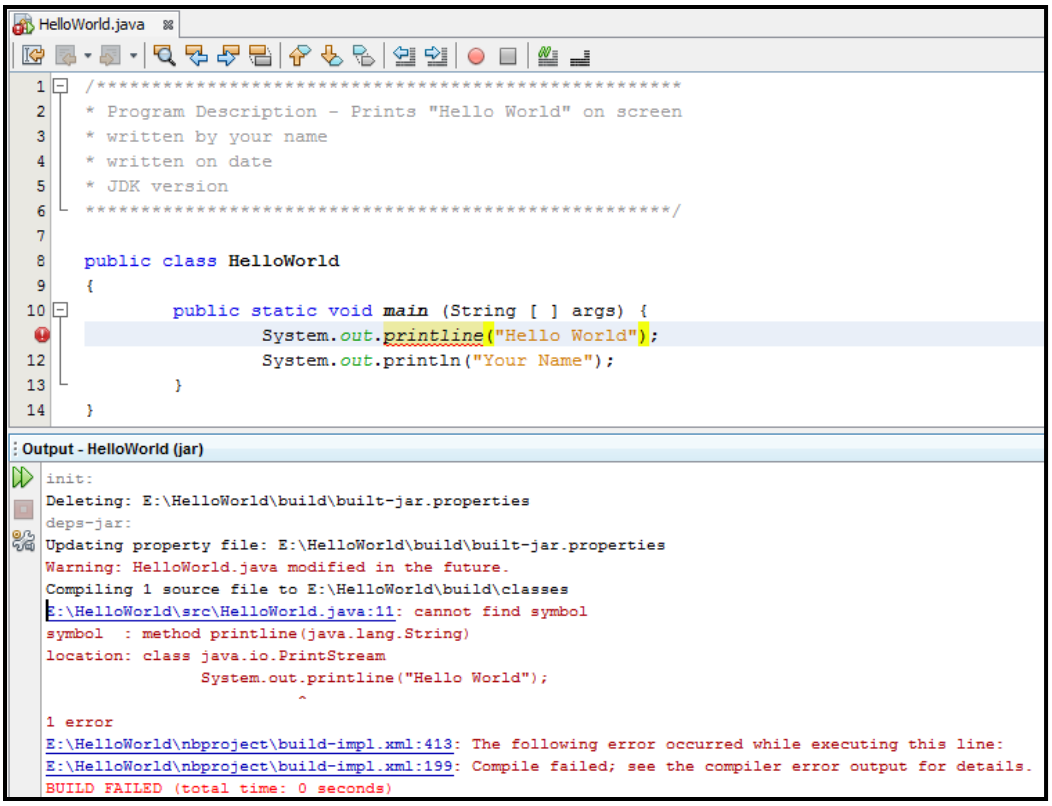
18. Open your code, by expanding the **HelloWorld project folder**, then expanding the **Source Package folder**, then expanding the **default package folder** *(this is default since we did not name this folder when we created the project)* as shown below. You will need to double click on the **HelloWorld.java** file to open the code.



19. You will now purposely make some errors in your program. Change the spelling of **println** to be **printline** as follows:

    System.out.printline("Hello World");

20. Compile the program by clicking on **Run** menu and choosing **Build Project**. You should get an error at the bottom of your screen as follows:

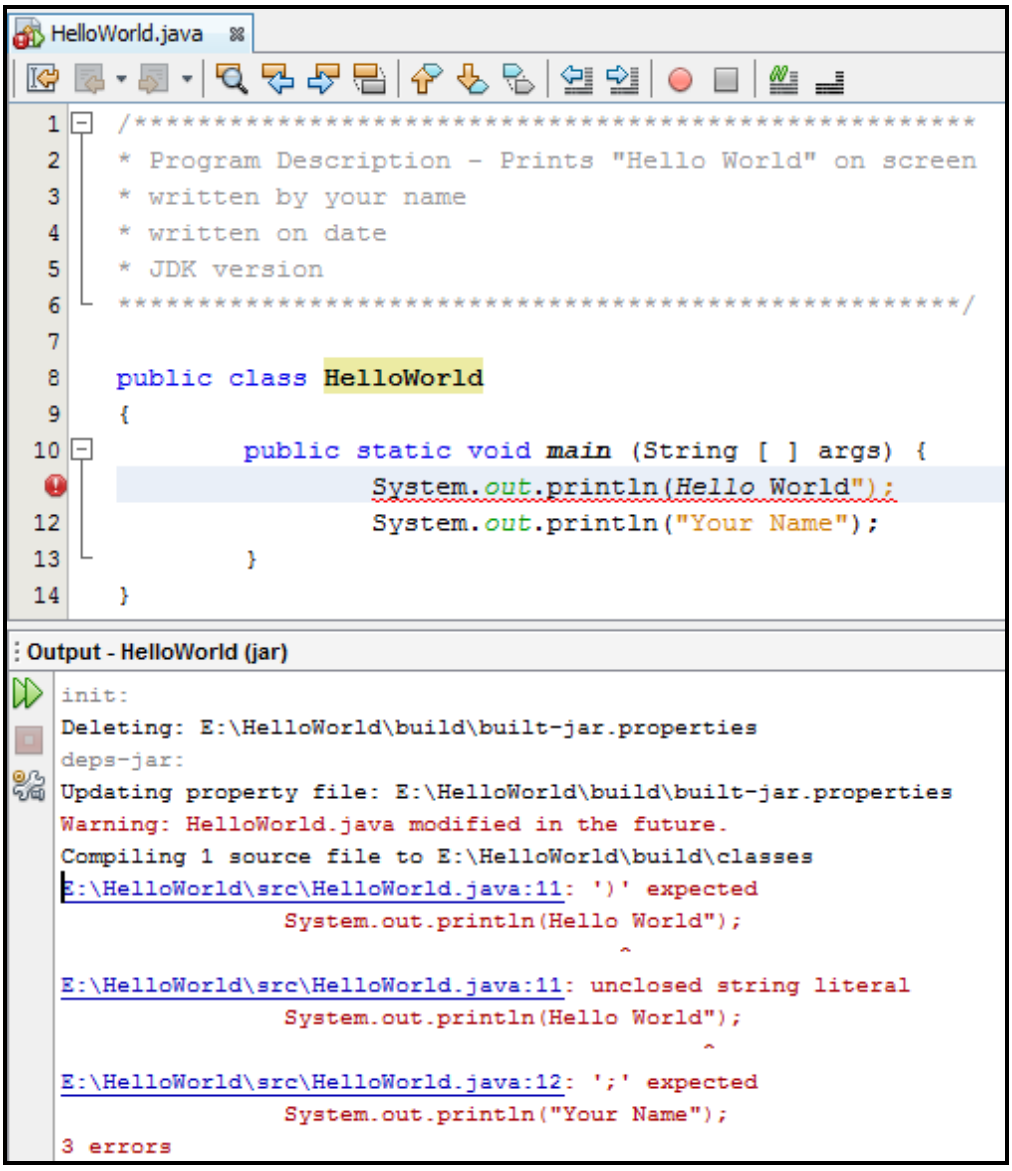


21. It is telling you that there is an error on line 11. It could not find a method spelled as **printline**. Some errors will be obvious and those are the nice ones to solve. Change the word **printline** to be **println** so that this error is corrected.

22. Compile the program. You should be rid of all errors and it should say BUILD SUCCESSFUL. Lesson learned: Names must be spelled exactly as Java expects.

23. Change line 11 by deleting the opening set of double quotes around Hello World as follows: **System.out.println(Hello World");**

24. Compile the program. You should get 3 errors at the bottom of your screen:



25. So, why would you get multiple error messages when you just made one mistake? This can happen. You may even get 15 errors for just one mistake. It really depends upon what mistake you make. In this case, it is saying it doesn't understand the Hello World and is saying it thinks it needs a parenthesis. This is not really the case. What it needs is a string enclosed in double quotes, but the error it shows is not real helpful. Now, insert the double quote back in front of the word *Hello*.

26. Compile the program. You should be rid of all errors. Lesson learned: The Java compiler doesn't always pinpoint the exact error. You must learn to look for errors anywhere on that line or previous line.

27. Not all errors are compilation errors. We now have a bug-free (no errors) Java program.

28. A programmer sometimes makes logic errors. Change line 11 to say:
    System.out.println("Helo Warld");
    Note: Hello and World are incorrectly spelled.

29. Compile the program. The program should get a compile with BUILD SUCCESSFUL. However, when this program is executed (RUN menu, then RUN MAIN PROJECT), you will not get the words displayed that you wanted. The reason that you get no errors in the compilation is because the words inside of double quotes can be anything. The computer has no idea what you are trying to accomplish. It will display anything that is in double quotes and doesn't check that part for incorrect spellings.

30. Please fix all of your errors and recompile. NetBeans automatically saves files when your project is compiled.

31. Let's adjust your **println** statements to use the **print** method instead of **println**. You should see that our text is all on one line. This doesn't look as nice as before.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.print("Hello World.");
        System.out.print("Your name.");
    }
}
```

```
Output - HelloWorld (run)
    run:
    Hello World.Your name.BUILD SUCCESSFUL (total time: 0 seconds)
```

32. Let's fix this by taking out the second **print** statement and adding a **\n** escape code to get your name to print on a new line.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.print("Hello World. \nYour name.");
    }
}
```

```
Output - HelloWorld (run)
    run:
    Hello World.
    Your name.BUILD SUCCESSFUL (total time: 0 seconds)
```

33. It might look better if we changed our **print** statement to a **println** statement so that the build information is on a new line.

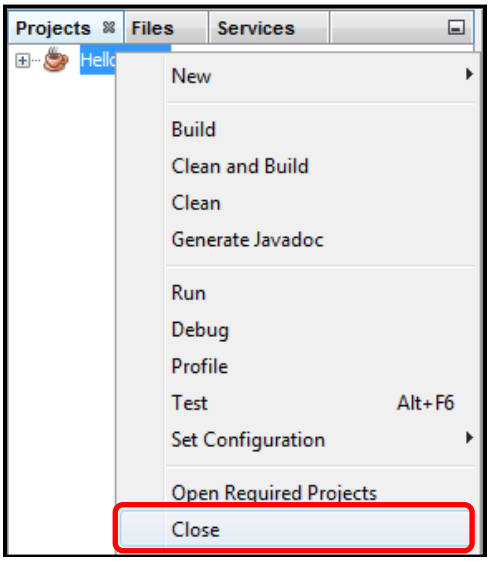
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World. \nYour name.");
    }
}
```

```
Output - HelloWorld (run)
    run:
    Hello World.
    Your name.
    BUILD SUCCESSFUL (total time: 0 seconds)
```

You can also put the code that prints your name on a separate line of code, but it won't change your result in the output window.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World. "
                + "\nYour name.");
    }
}
```

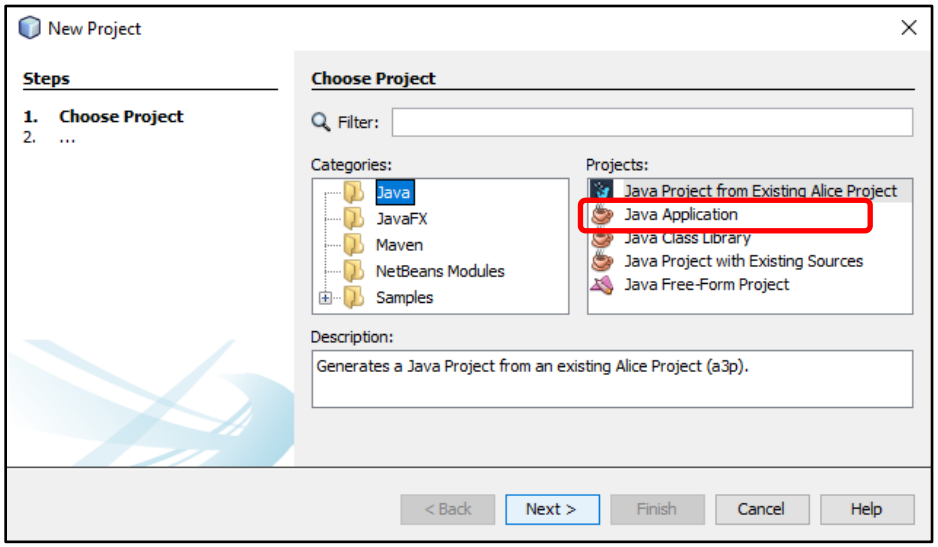
34. Close the HelloWorld project by right clicking on the project in the left pane as shown below:



Exercise 2: Drawing a Triangle Shape

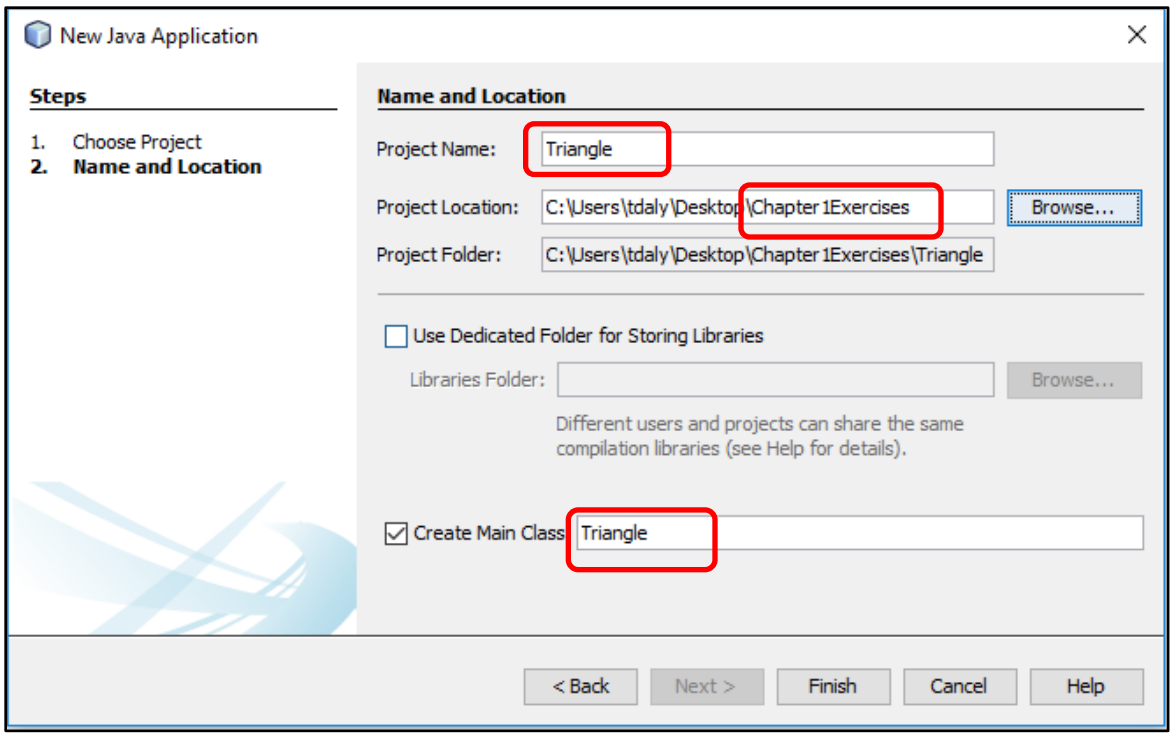
1. Open up the NetBeans environment.

2. Select the **File** menu and then choose **New Project**. Then choose **Java Application** as shown below.



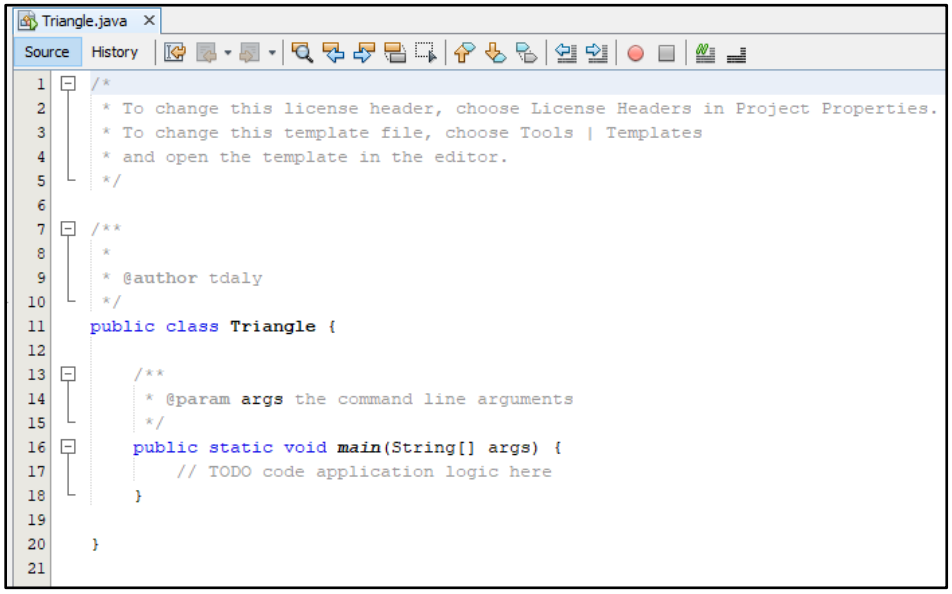
3. Click **Next**. Name your NetBeans project, select the **location** of where you would like to save your file, give your file *(Main Class)* a name *(make this name the same as your project name)*, and click **finish**. Although it is not necessary, we are going to name our projects and Java files *(main class)* have the same name. Therefore, make sure that the top and bottom boxes have the same name. NetBeans automatically will try to name your file (main class) triangle.Triangle. Erase the **triangle.** that NetBeans inserts before your file name. Make sure it looks like the following screenshot. Capitalization is important.

   *Project the name **Triangle** (no spaces)*
   *Main class name of **Triangle** (no spaces)*
   *Select the **Location** of where you would like to save your NetBeans project*

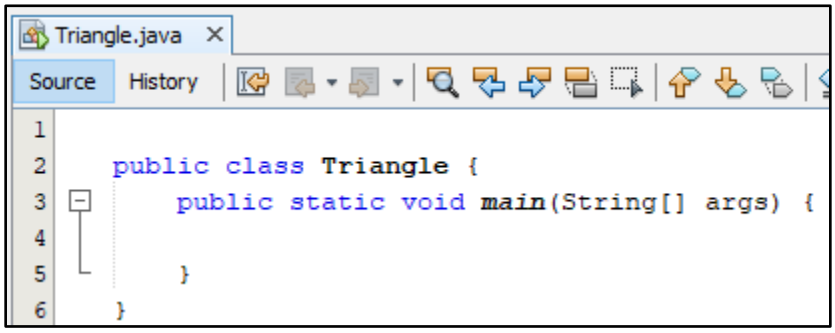


4. If line numbers are not showing, click **View** from the menu, then **Show Line Numbers.**

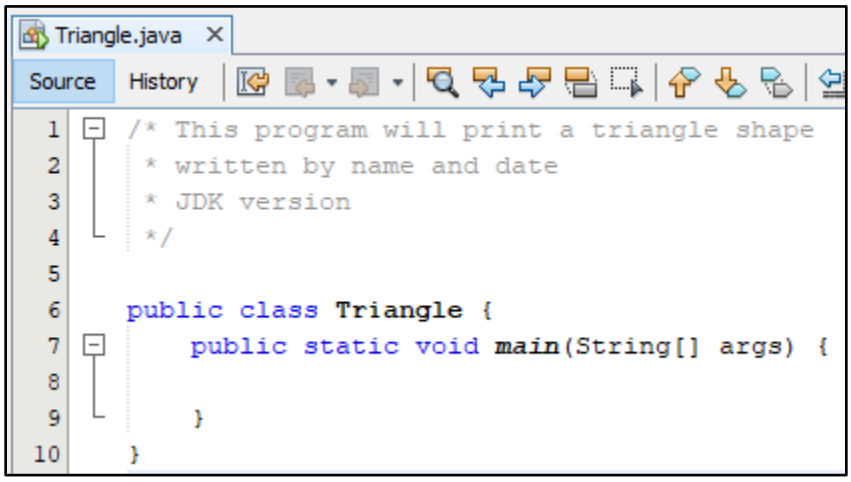
5. Your project should look as follows. HelloWorld.java is the file that we will be working with. *(Note: If your code has a package statement on line 5, then exit NetBeans and delete the project folder and do step 4 again. Make sure it looks like the screen shot provided. Look carefully at the textbox next to the Create Main Class label in the New Java Application dialog. For your information a package in Java is a folder. The package line indicates that you put your file in a folder when you created the project.)*



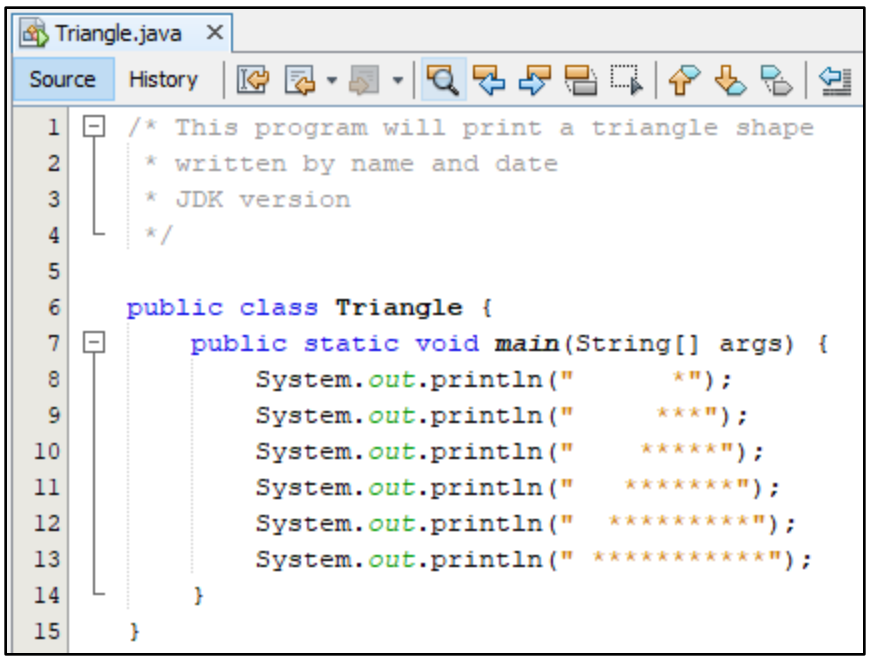
6. Let's set up our program by deleting the current program comments as shown below.



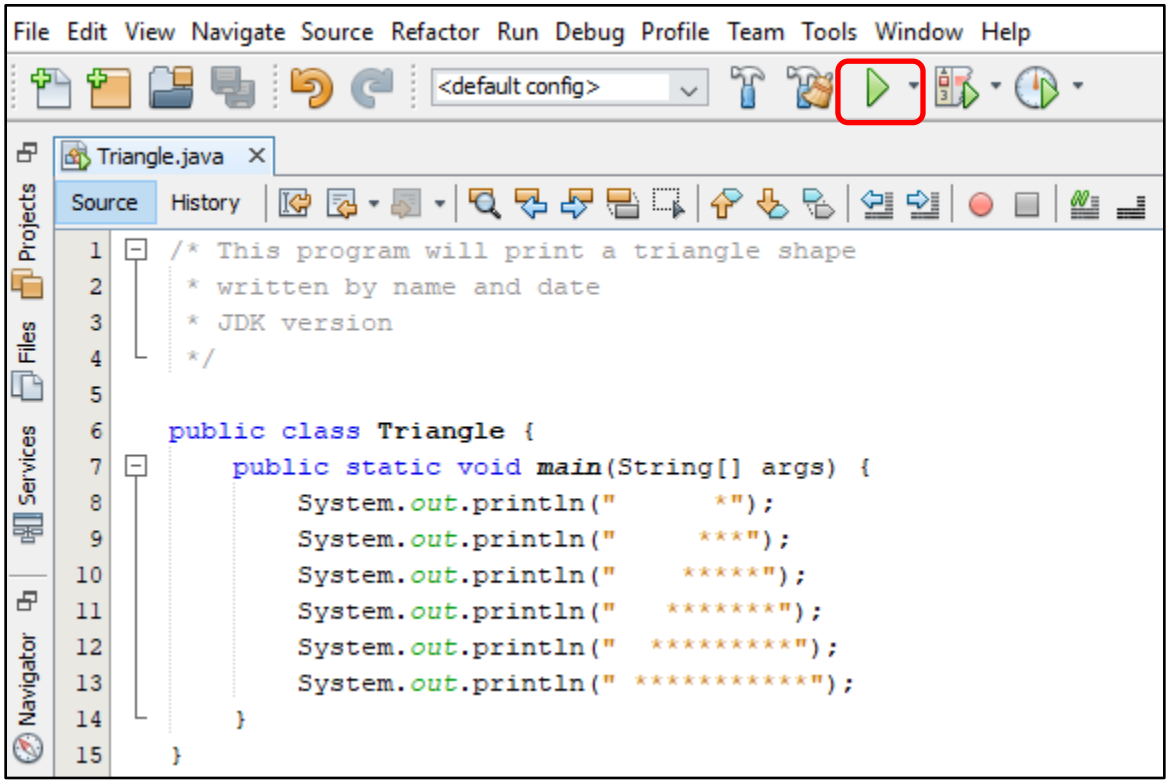
7. Now we will add our comments (documentation) to the top of our program. We will add our name, today's date, a description of the program, and the JDK version to the top of all of our programs. *(You can check to see what version of the JDK that you have by clicking **Help** from the menu and then **About** in the NetBeans environment. It will have **Java:** and then a number, this is your JDK version. You do not need the number after the underscore).*



8. Next, we need to add the Java code that will print our triangle shape. We will use **spaces** and the ***** symbol as text in our **println** method to draw our shape. Please add the following **println** statements to your main method.



9. Finally, we need to test our program to ensure that the triangle shape looks good in our output window. Run the program, by clicking on the **Run** button. You shouldn't have any red underlines in your program code. If you do, you will need to compare your code with the code shown below to determine what is different that is causing the problem. Code highlighted with red underlines are known as syntax errors.



10. Your output should look as follows.

```
run:
      *
     ***
    *****
   *******
  *********
 ***********
BUILD SUCCESSFUL (total time: 0 seconds)
```

Note: When you run your program in NetBeans, it should save the files automatically, but if you are unsure you can save. **DO NOT use the "Save as" option** in NetBeans. You can go to **File** and choose **Save** or you can use the shortcut command to save (Windows: ctrl + s, Mac: command + s). You can tell if the file is saved by looking at the tabs. If the filename is bold, then it has not been saved recently. If it is not bold, then it is saved. When you close NetBeans, it should warn you if you haven't saved. Please do not right click and rename Java files. Renaming Java files, will mess up the structure of your NetBeans projects and they won't work correctly. This is why I recommend you stay away from the "Save As" option; if you change the name or the location of your Java files, they won't work properly next time you open them.

11. This could also be written using the **\n** escape code and it would have the same output result. Either way of coding is fine. *Note: if you are joining two strings on separate lines, it doesn't matter which line you put the plus sign for the concatenation of the strings. You cannot have two plus signs in a row.*

```
/* This program will print a triangle shape
 * Name and Date
 * JDK Version
 */

public class Triangle {
    public static void main(String[] args) {
        System.out.println(  "       *" +
                           "\n      ***" +
                           "\n     *****" +
                           "\n    *******" +
                           "\n   *********" +
                           "\n ***********");
    }
}
```

Or you can write the code with the concatenation operator (plus sign) on the other line:

```
/* This program will print a triangle shape
 * Name and Date
 * JDK Version
 */

public class Triangle {
    public static void main(String[] args) {
        System.out.println("       *"
                         + "\n      ***"
                         + "\n     *****"
                         + "\n    *******"
                         + "\n   *********"
                         + "\n ***********");
    }
}
```

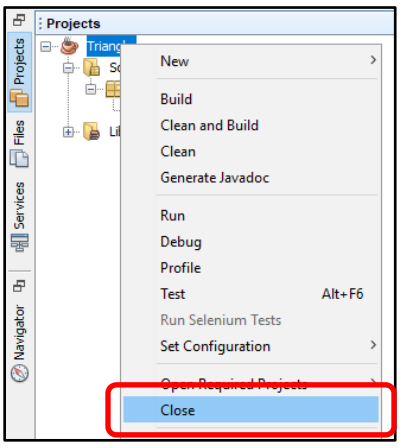
You cannot have the concatenation operator (plus sign) twice in a row.

```
1  /* This program will print a triangle shape
2   * Name and Date
3   * JDK Version
4   */
5
6  public class Triangle {
7      public static void main(String[] args) {
8          System.out.println("       *" +
                            + "\n      ***" +
                            + "\n     *****" +
                            + "\n    *******" +
                            + "\n   *********" +
                            + "\n ***********");
14     }
15 }
```

You will get an error that states *bad operand type String for unary operator '+'*

If you hover over the exclamation mark on the line numbers, you can see the errors without buiding the project.

12. Close the **Triangle** project by right clicking on the project in the left pane and choosing **Close**.



Summary

- Line by line instructions that tell a computer how to perform a task are called a **computer program**.
- A programming language that is written at the very low technical circuitry level of the computer is called a **low-level programming language**.
- **High-level programming languages** allow programmers to write programs using English terms.
- James Gosling at Sun Microsystems is credited with creating Java programming language. It was brought to the public in 1995.
- Java is an object-oriented programming language that is platform-independent. The Java slogan is "You can write once and run anywhere." The compiler creates a bytecode file (with class extension) that can be used on all types of computer systems (MAC, Linux, Windows, etc.) Bytecode is then fed to a Java Virtual Machine (JVM) where they are interpreted and executed.
- Computer languages each have their own **syntax**, or rules of the language.
- When you write a program, indenting is important so that you and other humans can understand your program.
- In every program that you write, there should be comment lines at the beginning of program. There should be a description of the program, author (you), date, and the JDK used for the program.
- Compilation error messages try to pinpoint the problem in your program but they are not always helpful. You may have to look around for the error or correct just the errors that you do understand and then compile again. You will get better at doing this and understanding this with additional experience.
- If your program is compiling fine, there still is no guarantee that it will work. An error-free compilation program only means that the Java compiler understands your commands, but the commands may not do what you want.
- The curly braces in Java are crucial. The symbols { } are generally found to the right of the "P" key.
- Programming can be very frustrating, but it also can be rewarding when you succeed.

Review Questions

1. JDK stands for:
    a. Java Details Kit
    b. Java Development Kit
    c. Java Decoder Kit
    d. Java Debugger Kit

2. All programming languages work on the Internet.
    a. True
    b. False

3. Java and JavaScript are the same programming language.
    a. True
    b. False

4. There are many high-level programming languages for computers.
    a. True
    b. False

5. The rules of a programming language are its _____.
    a. Vocabulary
    b. Syntax
    c. Logic
    d. Flowchart

6. Arguments to methods appear within
    a. Parentheses
    b. Semicolons
    c. Curly braces
    d. Quotation marks

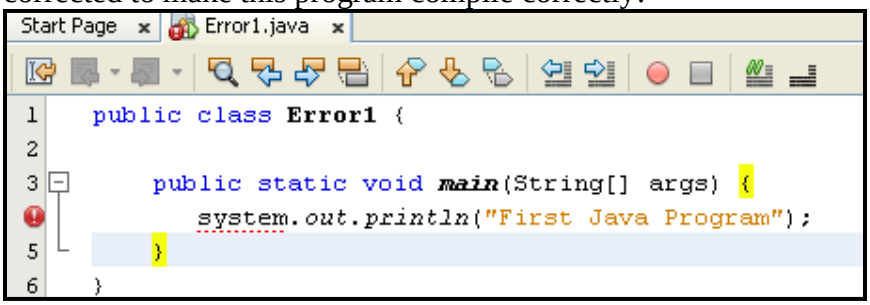
7. All Java application programs must have a method called _____.
    a. hello
    b. system
    c. main
    d. Java

8. Non-executing program statements that provide documentation to humans are called _____.
    a. Notes
    b. Classes
    c. Commands
    d. Comments

9. Once a program has compiled without errors, it will always execute perfectly.
    a. True
    b. False

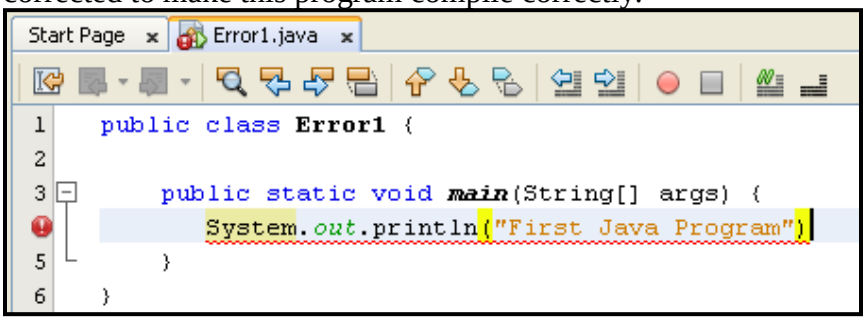
10. A computer _____ tells a computer how to perform a task.
    a. Switch
    b. Program
    c. Interface
    d. Guide

11. Look at the Java program at the top of the following illustration. This Java program was compiled and the compilation errors appear at bottom of screen. What needs to be corrected to make this program compile correctly?



    a. Change line 1 to say ***FirstJavaProgram*** instead of ***Error1***
    b. Change line 3 to say ***first*** instead of ***main***
    c. Change line 4 to say **System** instead of **system**
    d. Change line 4 to say "***Hello World***" instead of "***First Java Program***"

12. Look at the Java program at the top of the following illustration. This Java program was compiled and the Compilation errors appear at bottom of screen. What needs to be corrected to make this program compile correctly.



    a. Change line 1 to have semicolon at end of it.
    b. Change line 3 to have semicolon at end of it.
    c. Change line 4 to have semicolon at end of it.
    d. Change line 5 to have semicolon at end of it.

Solutions: *1) b 2) b 3) b 4) a 5) b 6) a 7) c 8) d 9) b 10) b 11) c 12) c*

Assignments

1-1 Printing Initials: Your goal is to print your initials in block letters

- ***Analyze and understand the problem to be solved.*** Display your initials in block letters.

- ***Develop the logic to solve the program.*** We can graph the block letters on graph paper, so we can see the spacing. The letters should be about 7 by 7 characters wide and have 5 spaces in between letters.

  Below is a blank grid so that you can draw your initials.

  Sample solution for initials YN (your name). You should not use YN unless your initials are YN.

Y							**Y**						**N**						**N**
	Y					**Y**							**N**	**N**					**N**
		Y			**Y**								**N**		**N**				**N**
			Y										**N**			**N**			**N**
			Y										**N**				**N**		**N**
			Y										**N**					**N**	**N**
			Y										**N**						**N**

- ***Code the solution in a programming language.*** We will need to code 7 print statements to accomplish the above graphic of the "YN" block letters. The spacing in these *System.out.println* statements must be perfect. Enter the coded solution into a new project in NetBeans. Name this project **Initials**.

- ***Test the program.***

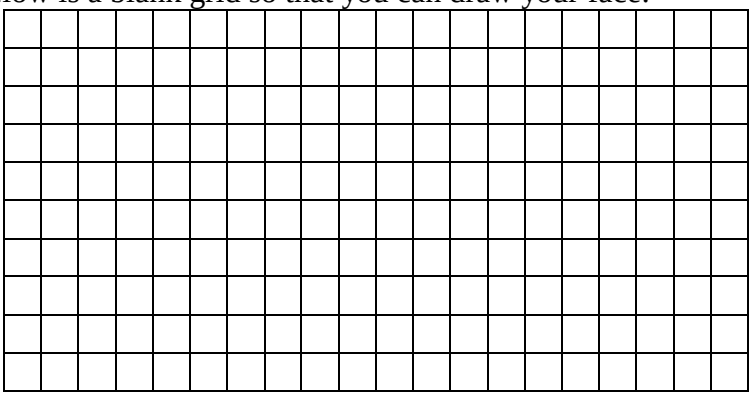
```
Output - Initials (run)
run:
Y     Y     N     N
 Y   Y      NN    N
  Y Y       N N   N
   Y        N  N  N
   Y        N   NN
   Y        N     N
BUILD SUCCESSFUL (total time: 2 seconds)
```

1-2 **Drawing a Face:** Your goal is to draw an ASCII art face. You can use any symbol that you want to make your drawing (*, -, etc.) and your face can be any emotion that you want (smiley, sad, surprised, etc.).

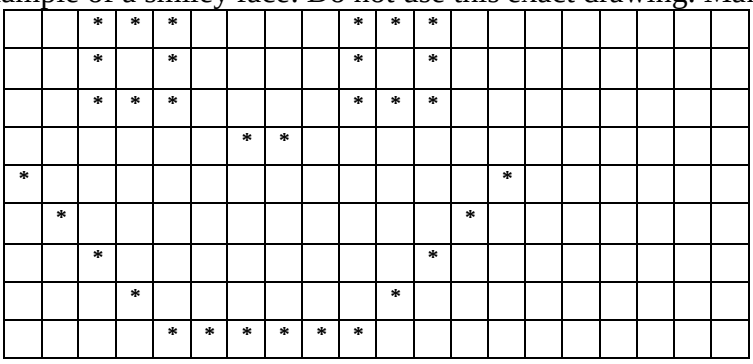
- ***Analyze and understand the problem to be solved.*** Draw a face using ASCII art.

- ***Develop the logic to solve the program.*** We can draw the face on graph paper, so we can see the spacing.

  Below is a blank grid so that you can draw your face.



  Example of a smiley face. Do not use this exact drawing. Make your own.



- ***Code the solution in a programming language.*** We will need to code print statements to accomplish the above drawing. The spacing in these *System.out.println* statements must be perfect. Enter the coded solution into a new project in NetBeans. Name this project **Emotion**.

- ***Test the program.***