Lecture 5

Object-Oriented Programming in Java

Aleksandar Stefanovski

CSCI 053
Department of Computer Science
The George Washington University
Spring, 2010

Learning Goals

Introduce Eclipse

Create objects in Java

Introduce variables as object references

Invoke methods on objects in Java

Create a method in Java

Pass a parameter to a method in Java

Introduce subclasses

Using Turtles in Java

We will work with Turtles in a World in Java

We have to define what we mean by a Turtle to the computer

We do this by writing a Turtle class definition

`Turtle.java`

We compile it to convert it into something the computer can understand

- Bytes codes for a virtual machine

`Turtle.class`

History of Turtles

Seymour Papert at MIT in the 60s

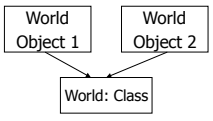
By teaching the computer to do something the kids are thinking about thinking

- Develop problem solving skills
- Learn by constructing and debugging something
  - Learn by making mistakes and fixing them

Creating Objects

To create objects we ask the object that defines the class to create it

- Each object is created in memory with space for the fields it needs
- Each object keeps a reference to the class that created it



The class is like a cookie cutter

- It knows how much space each object needs (shape)
- Many objects can be created from the class

Class as Object Factory

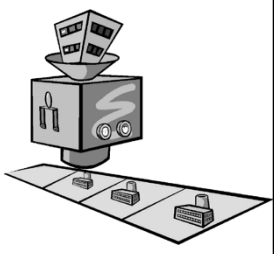
A class is like a factory that creates objects of that class

We ask a class to create an object by using the keyword:

`new` *`ClassName`*

We can also ask the class to initialize the object

- And pass data to help initialize it



Creating Objects in Java

In Java the syntax for creating an object is:

```
new Class(value, value, …);
```

Our Turtle objects live in a World object

- We must create a World object first
- Try typing the following in the interactions pane:

```
new World();
```

Creating Objects

If you just do

```
new World();
```

You will create a new World object and it will display

- But you will not have any way to refer to it again
- Once you close the window the object can be garbage collected
  - The memory that the object was using can be reused

We need a way to refer to the new object

- to be able to work with it again

Alice named your objects for you

- bunny, bunny2

Naming is Important

If you create a new contact in your cell phone you enter a phone number and a name

- Later you use the name to find the phone number

In programming we name things we want to refer to again

- Gives us a way to work with them
- Like the World object

In programming this is called declaring a variable

Declaring a Variable

To be able to refer to an object again we need to specify what type of thing it is and give it a name

- This is also called declaring a variable
- *Type name*; OR

```
Type name = new Class(value, value, …);
```

The equal sign doesn't mean equal

- But assign the value of the variable on the left to the result of the stuff on the right
- The following creates a variable named earth which refers to a World object created on the right
  - `World earth = new World();`

Declaring Variables

When you declare a variable the computer assigns memory to it

- The amount of memory depends on the type

For each variable the computer stores a map of the name to the memory location and the type

When you use the name the computer looks up the memory location

- And uses the value at that location

address	memory	
1	00000000	Object of type World
2	00001111	
3	00000000	
4	00111000	
5	00000000	
6	00111100	
7	01111000	
8	00000000	earth variable holds a reference to the World Object above
9	00000000	
10	00000000	
11	00000000	
12	00000001	

Limits on Declaring Variables

You can't declare two variables with the same name!

```
> World earth = new World();
> World earth = new World();
```

Error: Redefinition of 'earth'

You can change what an object variable refers to

```
> World earth = new World();
> earth = new World();
```

Cell Phones use Variables

In your cell phone you have names that map to phone numbers
- When you pick Home it looks up the number and uses it to make the call

You can't have two names that are exactly the same
- The phone wouldn't know which number you are referring to

You can modify the phone number for a name

- Barb 555-1235
- Home 555-2938
- Michael 555-3214
- Shannon 555-2921

Variables Sizes

What value(s) does the memory on the right represent?
- It could be 4 char values
  - 2 bytes each (16 bits)
  - Unicode
- Or 2 int values
  - 4 bytes each (32 bits)
  - 2's compliment
- Or 1 double value
  - 8 bytes each (64 bits)
  - In IEEE format
- Or an object reference
  - The size is up to the virtual machine

1 00001000
2 00100000
3 00000100
4 01000000
5 00000001
6 10000000
7 00111000
8 11110000

Declaring Variables and Creating Objects

You can declare a variable and assign it to refer to a new object in one statement

```
World earth1 = new World();
Turtle tommy = new Turtle(earth1);
```

Declaration of variables

Creating the objects

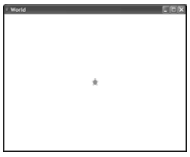
Turtle Basics

The world starts off with a size of 640 by 480
- With no turtles

```
World earth1 = new World();
```

The turtle starts off facing north and in the center of the world by default
- You must pass a World object when you create the Turtle object
  - Or you will get an error: java.lang.NoSuchMethodException: Turtle constructor

```
Turtle tommy = new Turtle(earth1);
```



Java Naming Conventions

Notice that we capitalize the names of the classes, but not the variable names
- This `World earth1 = new World();`
- is different than English
  - Capitalize proper nouns (the names of things)
  - Not the type of thing
    - Earth is a world.
    - Tommy is a turtle.

In Java it is the class names that are the most important
- Not the variable or method names

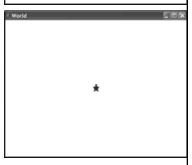
Creating Several Objects

You can create several World objects

```
World mars = new World();
```

You can create several Turtle objects

```
Turtle shar = new Turtle(mars);
Turtle jen = new Turtle(mars);
```



- One turtle is on top of the other

Moving a Turtle

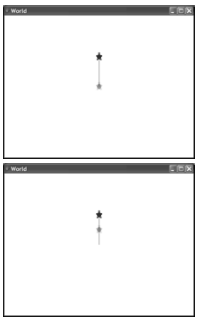
Turtles can move forward

```
jen.forward();
```

– The default is to move by
  - 100 steps (pixels)

You can also tell the turtle how far to move

```
shar.forward(50);
```



Turning a Turtle

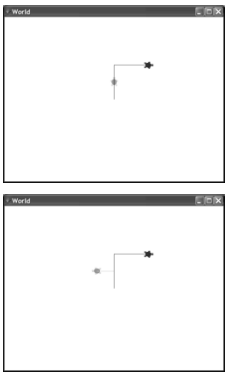
Turtles can turn

Right

```
jen.turnRight();
jen.forward();
```

Left

```
shar.turnLeft();
shar.forward(50);
```



Turning a Turtle

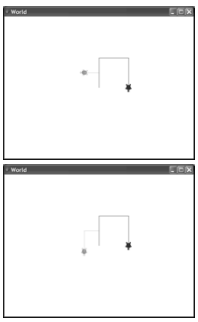
Turtles can turn by a specified amount

A positive number turns the turtle to the right

```
jen.turn(90);
jen.forward(100);
```

A negative number turns the turtle to the left

```
shar.turn(-90);
shar.forward(70);
```



The Pen

Each turtle has a pen

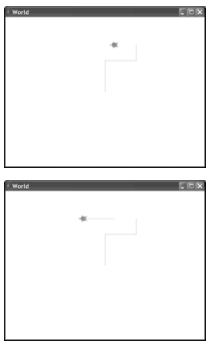
The default is to have the pen down to leave a trail

You can pick it up:

```
turtle1.penUp();
turtle1.turn(-90);
turtle1.forward(70);
```

You can put it down again:

```
turtle1.penDown();
turtle1.forward(100);
```



Drawing a Letter

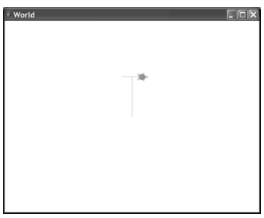
How would you use a turtle to draw a large letter T?

Process

- Create a World variable and a World object and a Turtle variable and object.
- Ask the Turtle object to go forward 100
- Ask the Turtle object to pick up the pen
- Ask the Turtle object to turn left
- Ask the Turtle object to go forward 25
- Ask the Turtle object to turn 180 degrees
- Ask the Turtle object to put down the pen
- Ask the Turtle object to go forward 50

Drawing a T

```
World world1 = new World();
Turtle turtle1 = new Turtle(world1);
turtle1.forward(100);
turtle1.penUp();
turtle1.turnLeft();
turtle1.forward(25);
turtle1.turn(180);
turtle1.penDown();
turtle1.forward(50);
```



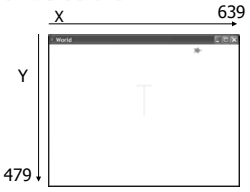
Moving to a Location

A turtle can move to a particular location

```
turtle1.penUp();
turtle1.moveTo(500,20);
```

Coordinates are given as x and y values

- X starts at 0 on the left and increases horizontally to the right
- Y starts at 0 at the top of the window and increases to the bottom
- A new turtle starts out at 320,240 by default



Challenge

Create a World object

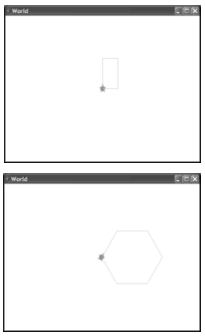
- Don't forget to declare a variable to hold a reference to it

Create a turtle object

- Don't forget to declare a variable to hold a reference to it

Use the turtle to draw a

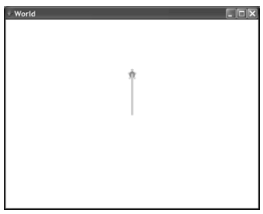
- Rectangle (but, not a square)
- Equilateral triangle
- Hexagon



Setting the Pen Width

You can change the width of the trail the pen leaves

```
World world1 = new World();
Turtle turtle1 = new Turtle(world1);
turtle1.setPenWidth(5);
turtle1.forward(100);
```



Setting the Pen Color

Use `setPenColor` to set the color of the pen

```
turtle1.setPenColor(java.awt.Color.RED);
```

There are several predefined colors

- In the package `java.awt`
  - A package is a group of related classes
- In the class Color

To use them you can use the full name

- `java.awt.Color.RED`

Setting Colors

You can change the pen color

```
turtle.setPenColor(java.awt.Color.RED);
```

You can change the turtle color

```
turtle1.setColor(java.awt.Color.BLUE);
```

You can change the turtle's body color

```
turtle1.setBodyColor(java.awt.Color.CYAN);
```

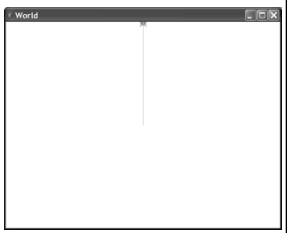
You can change the turtle's shell color

```
turtle1.setShellColor(java.awt.Color.RED);
```

Objects can Refuse

Turtles won't move completely out of the boundaries of the world

```
World world2 = new World();
Turtle turtle2 = new Turtle(world2);
turtle2.forward(600);
```



Objects send Messages

Objects don't "tell" each other what to do
- They "ask" each other to do things

Objects can refuse to do what they are asked
- The object must protect it's data
  - Not let it get into an incorrect state
  - A bank account object shouldn't let you withdraw more money that you have in the account

Creating Methods

In Alice you could create a method
- like teaching a bunny to hop

Creating a Method

We can name a block of Java statements and then execute them again
- By declaring a method in a class

The syntax for declaring a method is
- *visibility returnType name*(*parameterList*)
- Visibility determines access
  - Usually public or private
  - The return type is the type of thing returned
  - If nothing is returned use the keyword **void**
- Name the method starting with a lowercase word and uppercasing the first letter of each additional word

Example Method

```
public void drawSquare
  ()
  {
    this.turnRight();
    this.forward(30);
    this.turnRight();
    this.forward(30);
    this.turnRight();
    this.forward(30);
    this.turnRight();
    this.forward(30);
  }
```

- The visibility is **public**
- The keyword **void** means this method doesn't return a value
- The method name is **drawSquare**
- There are no parameters
  - Notice that the parentheses are still required
- The keyword **this** means the object this method was invoked on

Adding a Method to a Class

1. Open file Turtle.java
2. Type the method before the last }
3. Compile open files

Try the New Method

Compiling resets the interactions pane

Clearing all variables
- But you can still use the up arrow to pull up previous statements

You will need to create a world and turtle again

```
World world1 = new World();
Turtle turtle1 = new Turtle(world1);
turtle1.forward(50);
turtle1.drawSquare();
turtle1.turn(30);
turtle1.drawSquare();
```

Adding a Main Method

In Alice there is a way to specify what method to execute when the world starts

In Java when you execute a class you can have a main method
- This is where execution will start

Main Method Syntax

The main method is a class (static) method that can take an array of strings
- It is a class (static) method since no objects of the class exist yet

```
public static void main(String[] args)
{
  // statements to execute
}
```

Better Method to Draw a Square

A method to draw a square

```
public void drawSquare()
  {
    int width = 30;
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
  }
```

- We added a local variable for the width
  - Only known inside the method
- This makes it easier to change the width of the square
- But, we still have to recompile to draw a different size square

Testing the Better Method

Test with:

```
public static void main(String[] args)
  {
   World world1 = new World();
   Turtle turtle1 = new Turtle(world1);
   turtle1.forward(50);
   turtle1.drawSquare();
   turtle1.turn(30);
   turtle1.drawSquare();
 }
```

Passing a Parameter

```
public void drawSquare(int width)
  {
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
  }
```

- Parameter lists specify the type of thing passed and a name to use to refer to the value in the method
- The type of this parameter is **int**
  - For integer
- The name is **width**
- Values are passed by making a copy of the passed value

Testing with a Parameter

Test a method with a parameter

```
public static void main(String[] args) {
   World world1 = new World();
   Turtle turtle1 = new Turtle(world1);
   Turtle turtle2 = new Turtle(world1);
   turtle1.forward(50);
   turtle1.drawSquare(30);
   turtle2.turn(30);
   turtle2.drawSquare(50);
}
```

How Does That Work?

When you ask `turtle1` to `drawSquare(30)`

```
turtle1.drawSquare(30);
```

- It will ask the Turtle Class if it has a method `drawSquare` that takes an int value
  - And start executing that method
  - The parameter width will have the value of 30 during the executing of the method
  - The `this` keyword refers to turtle1

When you ask `turtle2` to `drawSquare(50)`

```
turtle2.drawSquare(50);
```

- The width will have a value of 50
- The `this` refers to turtle2 (the object the method was invoked on)

Challenges

Create a method for drawing a rectangle
- Pass the width and height

Create a method for drawing an equilateral triangle
- all sides have the same length
- Pass in the length

Create a method for drawing a house
- Using the other methods

Create a method for drawing a school
- Using the other methods

Subclasses in Java

The Turtle class is a subclass of `SimpleTurtle`
- public class Turtle *extends* SimpleTurtle

This means it inherits methods and fields from `SimpleTurtle`
- See if you can find the forward and `turnRight` methods in `SimpleTurtle`

Classes can subclass another class in Java and Alice

Summary

You can create objects from classes in Alice and Java

Each object needs a unique way to refer to it
- In Java we call this declaring a variable

You can create new methods
- *visibility returnType name*(*Type name*, *Type name*, …)
- Let's you reuse a block of statements

You can pass parameters to methods
- To make them more flexible and reusable

You can create subclasses of other classes
- They will inherit fields and methods from the parent class