Sunday, December 30, 2018 1:57 PM

Objects

- •Strings group a sequence of individual characters and let us compare to other Strings, count its characters, convert it to uppercase, etc.
- •Arrays and ArrayLists let us collect multiple items of the same type then loop over the data and do the same thing to each one easily.

So far we've been writing code that always executes based on the main method, moving from top to bottom- this is called "procedural programming".
In this module, we're going to define new entities in Java that can interact with one another, this is called "object oriented programming".
Java has a good number of object oriented features, which is why it is a popular language to learn in. However, plenty of different languages have these features such as C#, C++ or Python.

Why Objects

By now you should be familiar with creating variables. A variable creates a space in memory for a piece of data. In the case of primitives, this is a very small piece of data. Sometimes you want to group pieces of data together, for example an Array groups a series of ordered data of the same type.
As you are working in code you might run into situations where there are specific groupings of different types of data- but up until now you haven't had a way to tell the computer these variables are associated.
For example, let's say you are trying to write a program to manage student information.

```
String stu1Name = “Ada Lovelace”;
int stu1Grad = 2017;
int stu1ID = 12345;
double stu1GPA = 3.2;
int stu1Abs = 4;

String stu2Name = “Alan Turing”;
int stu2Grad = 2016;
int stu2ID = 12346;
double stu2GPA = 3.8;
int stu2Abs = 9;
```

```
String stu3Name = “Margret Hamilton”;
int stu3Grad = 2017;
int stu3ID = 12347;
double stu3GPA = 3.92;
int stu3Abs = 0;
```

As you can see, this can get pretty unmanageable very quickly. When you create your own object, you can formally group this data together under a brand new Data Type. This way all this information can be stored under a single variable!

```
public class WhosGraduating {
   public static void main (String[] args) {
      Student ada = new Student("Ada Lovelace", 2017, 12345, 3.2, 4);
      Student alan = new Student("Alan Turing", 2016, 12346, 3.8, 9);
      Student margret = new Student("Margret Hamilton", 2017, 12347, 3.92, 0);
   }
}
```

Creating objects lets you organize your code into more modular units, which can be extremely useful when writing large amounts of code. You can still accomplish the same things with procedural programming, but it can be much more difficult to manage the code base.There are still plenty of situations in which procedural programming is used, especially as we write code for devices with limited computing resources, however if you intend to work as a software engineer you will likely be working with a number of people in one large code base- this is where object oriented programming makes everyone's lives easier!

Object Definitions

In order to create a new object you have to write a class that "defines" it. In this way you can think of this as adding a new keyword to Java, but to do so you have to tell Java what this keyword means. Here's how we would define a "Student" object that includes all the variables from the example above:

```
public class Student {
   String name;
   int grad;
   int ID;
   double GPA;
   int abs;

   public Student(String name, int grad, int ID, double GPA, int abs) {
      this.name = name;
      this.grad = grad;
      this.ID = ID;
```

```
        this.GPA = GPA;
        this.abs = abs;
    }

    public boolean isGraduating() {
        return (GPA > 2.0 && abs < 10 && grad == 12);
    }
}
```

Notice that this class does not have a main method. That is because this code cannot be executed by itself, but must instead be invoked by a different class that does have a main method.

Objects often model objects in the real world and their characteristics can help us design them. For example, a bike shop might define their Bicycle object to include data like its brand name, model name, list price, tire size, number of gears, whether it has been sold or not, the sales price, and more. The shop might need to do things like change the Bicycle's price when there's a sale, mark it as sold and more.

This definition of a Bicycle object is contained within a class, just like all the code we've been writing. However, this is a special class that tells Java how to create individual occurrences or "instances" of your design. In our Bike Shope example each instance of "Bicycle" would represent a real bike in the shop. Our definition acts as a "blue print" dictating what data and behavior each Bicycle should have, then we can create different instances of Bicycles each with its own values for the data we defined.

)

Once we have defined what a Bicycle object should look like in Bicycle.java then we can create individual copies of Bicycles like we have been doing with the objects we already use:

```
Bicycle bike1 = new Bicycle();
```

This code looks at what you have done in Bicycle.java and creates an object you can actually use. The distinction between the class definition and the objects that are created based on the definition is easy to misunderstand. Here's a summary of which is which:

class

- Defined in Object class
- Specifies types of "attributes" that all objects of this type have
- Specifies "behaviors" of its objects in methods that all objects of this type have
- Java program can contain multiple different Object definitions

object (instance)

- Created when program runs using "new" operator
- Holds individual values of attributes that can change as program runs
- Executes behavior on its personal set of attribute values
- When a Java program is run it can create as many different copies of each Object that is defined