Revised Activity 1.1.2 Your First Class

Otakar Andrysek

Subtarget 1.1: Identify the three main components of a java file, the three most commonly used primitive variable types and the most common object variable type.

Introduction

When you program in an **object-oriented language**, you are programming with objects. An **object** describes a set of data in a program, and it can represent anything—a text message, a list of contacts, a bicycle—anything that needs to be represented. In this activity, you will write a simple Media Library application. You will create objects that represent your favorite songs, movies, and books.

Materials

- Computer with BlueJ ~~and Android™ Studio~~
- ~~Android™ tablet and USB cable, or a device emulator~~

Part I: Create a Song Class

1. As you did in the last *Activity 1.1.1 Intro to Android Development*, create a new project in BlueJ. Call the project **MediaLib** and create a new class called MediaLib.

   Done.

2. As you did in your HelloWorld project, delete the code that BlueJ auto-generated, and replace it with a main similar to the main in HelloWorld.

   Done.

3. Change the message that prints to "Welcome to your Media Library".

   Done.

Your Media Library will eventually contain lists of your favorite media items, such as names of songs, movies, and books. Each of these categories will be a class in your program.

4. Learn more about **Classes and Objects** in the online Java Review material
5. Observe your MediaLib class in the BlueJ editor. Read **FirstClass and SecondClass** up through the section titled **Running a Java Program**. Compare your MediaLib class with SecondClass. ~~What do they have in common?~~

   Both print a string, but SecondClass is written with a different name. Also, my main does not have arguments passed to it.

6. In BlueJ, click **New Class…** and enter the name **Song**. By convention, classes have names that begin with an uppercase letter, so be sure to capitalize properly. Click **Ok**.

   Done.

   This creates a new file called Song.java, which defines your new class. The file is automatically included in your MediaLib project.

7. Edit your Song class (double-click its box or select **Open Editor** from the context menu) and take a closer look at the code that BlueJ generated. After the line with public class Song that begins the class definition, you will see:

```
1     // instance variables - replace the example below with your own
2     private int x;
```

This indicates a **variable**. **Learn about variables in Java**. A **primitive** data type is a built-in part of Java language, without using external definitions such as your Song object.
What are the four main variable types you will use this quarter, ~~in both~~ and identify whether each is the primitive or object category?

Integer - Primitive
String - Object
Boolean - Primitive
Double - Primitive

The **int** variable x has been defined in your Song class and means that in addition to being an integer variable, it is also an **instance field** for your class. Instance fields contain the data for an object.

8. Read the section in the online resource titled **Fields - Instance Variables**. What are the instance fields for the Person class?

   Name and cell.

9. You will define two instance fields to represent the data for your Song class. The first is a rating on a scale of 1 to 10. Change the name of variable x to rating.

   Done.

10. Create a second data item for your Song class—the title of the song. This should be a **String** data type and also be private. You will learn much more about strings in later activities; for now, you only need to know they are just a sequence or "string" of characters.

After the instance fields, the next few lines in your Song class are:

```
1     /**
2      * Constructor for objects of class Song
3      */
4     public Song()
5     {
6         // initialise instance variables
7         x = 0;
8     }
```

This part of your class, specifically lines 4–8 above, is called the **constructor**.

11. **Read about constructors**. What is the role of a constructor?

Constructors are used to initialize member variables of a class (fields).

When you **initialize** a variable, you are assigning it a value for the first time. You may have noticed that your class no longer has an `x` instance field, but it does have a `rating` instance field.

12. In the constructor, change the line of code that initializes `x` to initialize `rating` instead.

    Done.

13. Add another line of code to your constructor so that the instance field `title` is initialized to a string without any characters:

    ```
    1      title = "";
    ```

    Notice there is no space between the quotation marks. This is called the **empty string** because it is a string without any characters.

    How does this work in respect to memory?

14. Finally, delete the rest of the auto-generated code, but be sure to leave the last curly brace that closes or finishes your class definition.

    Done.

15. Compile your code and fix any bugs you may have.

    Done.

Part II: Make a Song Object

Creating a new object creates a place in memory where information about the object is stored. In this part of the activity, you will learn how to create a new object and how to access the information associated with the object.

16. Continuing in MediaLib, create a Song using the new keyword to **instantiate**, or create, a new object from the Song class:

```
1     Song song1 = new Song();
```

The song1 variable is an **object reference**. An object reference refers to an object in memory; it points to a place in memory where the data for an object is stored. In this case, song1 points to where title and rating are stored as a Song object.

17. Use the following to print out the song1 variable:

```
1     System.out.println(song1);
```

What is shown?

Welcome to your Media Library
Song@4c20e0

When you use println to show the contents of the song1 variable, it shows the data type (Song) the @ sign, and then the reference or the address in memory of that object.

To show the actual data that the song1 variable points to is called **de-referencing** an object. You will de-reference an object in the next section.

Yay!

Part III: Create Methods

De-referencing an object means to access the information, or data, contained in the object. A common way to de-reference an object and *access* its data is to use an object **method** called an **accessor**. A common way to de-reference an object and *set or change* its data is to use a method called a **mutator**. In this part of the activity, you will discover how to get and set data in an object.

18. Read the section titled **Methods**. What could one accessor for your Song class do?

Set a Title

19. Write an accessor method for your Song class called getTitle().

```
1    public String getTitle() {
2        return title;
3    }
```

a. The above method is public, meaning other objects, including your MediaLib object can use or access this method. If you had written private, other classes would not be able to access this method.

b. After public, you specified String. This means that the getTitle() method will **return** or pass back some String value. Not all methods return something.

c. Then there is the contents or the body of your method. Here, the method simply returns the title of your song.

20. Now write a mutator method for your Song class called setTitle().

```
1    public void setTitle(String t) {
2        title = t;
3    }
```

a. The above method is also public, meaning other objects, including your MediaLib object can use or access this method.

b. After public, you specified void. This method returns nothing. Notice it does not have a return statement like your accessor does.

c. After the method name setTitle, you specified (String t). This means a string value is passed to your method and assigned to the variable t. Think of t as the input to your method.

d. Finally, you assigned the value of t to title.

21. With accessor and mutator methods defined, you can get and set the title of a song. In the main method of MediaLib, add:

```
1       song1.setTitle("Johnny B. Goode");
2       System.out.println(song1.getTitle());
```

22. In the same way you just created an accessor and a mutator for your title instance field, create an accessor and a mutator for the rating instance field. Show the rating of your song using System.out.println(…). Remember that rating is an int and not a String.

    This may seem like a lot of work just to manage a bit of data, but it keeps data secure and consistent. Suppose, for example, you want to keep all ratings between 1 and 10, or between 1 and 5. Your mutator can make sure that happens.

    Done!
    I'm really enjoying this so far.

Part IV: Create Movies and Books

Practice what you have learned by creating two new classes to keep track of movies and books. Refer to the code that is already written for the song class as an example.

23. In the same way you created a Song class, the instance fields, and the mutators and accessors, create a Movie class with the same instance fields, mutators, and accessors.

    Done. (I just duplicated the Song class and replaced song with movie, I hope this is all I'm supposed to do)

24. Repeat the process to create a Book class with the same instance fields and mutators and accessors.

    Done.

25. Create another Song object, as well as two Movie and two Book objects in the main method of MediaLib and show the data with: System.out.println(…);
    Paste a snip of your output here:

```
Welcome to your Media Library
Song@607f9200
Song@611191a1
Movie@7b937570
Movie@17583a58
Book@b2b4521
Book@665a517c
Johnny B. Goode
1
John Cena Theme
1
Interstellar
1
Forest Gump
1
Moby Dick
1
1984
1
```

Conclusion

1. What are the three main parts to a java file?

   Objects
   Classes
   Methods
   Instance Variables?

2. What do we call the two most common types of class methods?

   Accessors and Mutators

3. What would return if I evaluate this conditional:
   if (song1 == “Johnny B. Goode”) System.out.print(“I knew it!”);
   Why?

   Nothing would print because song1 is a memory address.
   This would work:
   if (song1.getTitle() == “Johnny B. Goode”) System.out.print(“I knew it!”);

4. There is a lot of duplication of code between your song, movie, and book classes. What instance fields and methods do they all have? Can you think of a way to reduce this duplication?

   They all have instance fields for title and rating. The methods are also copies of each other. Because these classes are all public I'm sure there is a more efficient way to code this, in my limited knowledge I have no idea how, hopefully I will learn!

   Update: I could have book, movie, song all be one class, and just add one other instance field: type. This variable would hold the content type, all in one class.

All my code for this year will be available on GitHub!
https://github.com/otakar-sst/CSA/tree/master/Java%20Programs/Lesson%201