COMP1618 Exercise 7 Classes and Method

Objectives

- Be able to declare a new class
- Be able to write a constructor
- Be able to write methods that return a value
- Be able to write methods that take arguments
- Be able to instantiate an object
- Be able to use calls to instance methods to access and change the state of an object

Deliverable

To create the following file with the given instruction below:

- Television.java

Introduction

Everyone is familiar with a television. It is the object we are going to create in this lab. First we need a blueprint. All manufacturers have the same basic elements in the televisions they produce as well as many options. We are going to work with a few basic elements that are common to all televisions. It has a brand name (i.e. it is made by a specific manufacturer). The television screen has a specific size. It has some basic controls. There is a control to turn the power on and off. There is a control to change the channel. There is also a control for the volume. At any point in time, the television's state can be described by how these controls are set.

We will write the television class. Each object that is created from the television class must be able to hold information about that instance of a television in fields. So a television object will have the following attributes:

• **manufacturer**. The manufacturer attribute will hold the brand name. This cannot change once the television is created, so will be a named constant.

• **screenSize**. The screenSize attribute will hold the size of the television screen. This cannot change once the television has been created so will be a named constant.

• **powerOn**. The powerOn attribute will hold the value true if the power is on and false if the power is off.

• **channel**. The channel attribute will hold the value of the station that the television is showing.

• **volume**. The volume attribute will hold a number value representing the loudness (0 being no sound).

These attributes become **fields** in our class.

The television object will also be able to control the state of its attributes. These controls become **methods** in our class.

• **setChannel**. The setChannel method will store the desired station in the channel field.

• **power**. The power method will toggle the power between on and off, changing the value stored in the powerOn field from true to false or from false to true.

• **increaseVolume**. The increaseVolume method will increase the value stored in the volume field by 1.

• **decreaseVolume**. The decreaseVolume method will decrease the value stored in the volume field by 1.

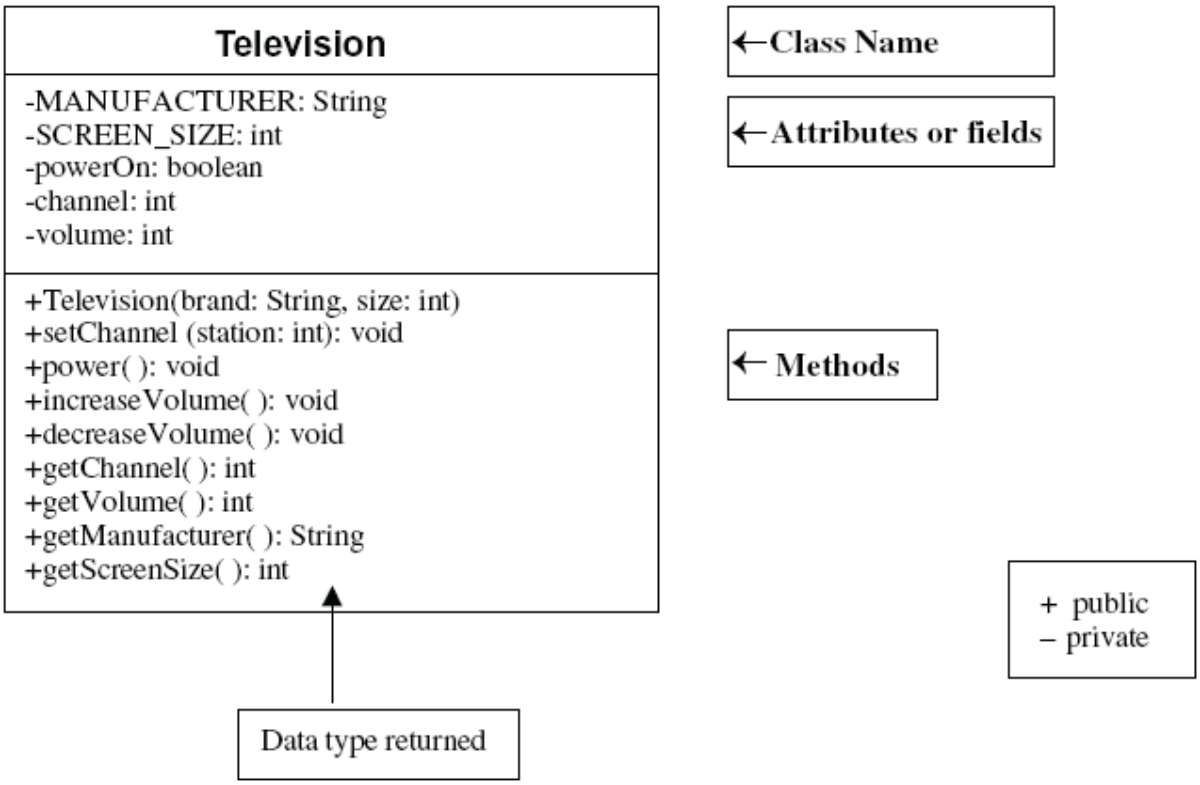
• **getChannel**. The getChannel method will return the value stored in the channel field.

• **getVolume**. The getVolume method will return the value stored in the volume field.

• **getManufacturer**. The getManufacturer method will return the constant value stored in the MANUFACTURER field.

• **getScreenSize**. The getScreenSize method will return the constant value stored in the SCREEN_SIZE field.

We will also need a constructor method that will be used to create an instance of a Television. These ideas can be brought together to form a UML (Unified Modelling Language) diagram for this class as shown below.



Task 1 Creating a New Class – All coding should be done in NetBeans

1. In a new NetBeans project, create a class definition called Television.
2. Put a program header (comments/documentation) at the top of the file
   // The purpose of this class is to model a television
3. Declare the 2 constant fields listed in the UML diagram.
4. Declare the 3 remaining fields listed in the UML diagram.
5. Write a comment for each field indicating what it represents.
6. Save this file as *Television.java*.

Task 2 Writing a Constructor

1. Create a constructor definition that has two parameters, a manufacturer's brand and a screen size. These parameters will bring in information.

2. Inside the constructor, assign the values taken in from the parameters to the corresponding fields.

3. Initialize the powerOn field to false (power is off), the volume to 20, and the channel to 2.

4. Write comments describing the purpose of the constructor above the method header.

5. Do not run.

Task 3 Methods

1. Define accessor methods called getVolume, getChannel, getManufacturer, and getScreenSize that return the value of the corresponding field.

2. Define a mutator method called setChannel accepts a value to be stored in the channel field.

3. Define a mutator method called power that changes the state from true to false or from false to true. This can be accomplished by using the NOT operator (!). If the Boolean variable powerOn is true, then !powerOn is false and vice versa. Use the assignment statement

   ```
   powerOn = !powerOn;
   ```

   to change the state of powerOn and then store it back into powerOn (remember assignment statements evaluate the right hand side first, then assign the result to the left hand side variable.

4. Define two mutator methods to change the volume. One method should be called increaseVolume and will increase the volume by 1. The other method should be called decreaseVolume and will decrease the volume by 1.

5. Write comments above each method header describing the purpose of the method.

6. Do not run.

Task 4 Running the Application

1. You can only execute (run) a program that has a main method, so there is a driver program that is already written to test out your Television class. Add the file *TelevisionDemo.java* to your NeBeans project.

2. Compile and run TelevisionDemo and follow the prompts.

3. If your output matches the output below, *Television.java* is complete and correct. You will not need to modify it further for this lab.

OUTPUT (boldface is user input)

A 55 inch Toshiba has been turned
on. What channel do you want? **56**
Channel: 56 Volume: 21
Too loud!! I am lowering the
volume. Channel: 56 Volume: 15

Task 5 Creating another Instance of a Television

1. Edit the *TelevisionDemo.java* file.

2. Add to the comment header as indicated at the top of the program.

3. Declare another Television object called portable.

4. Instantiate portable to be a Sharp 19 inch television.

5. Use a call to the power method to turn the power on.

6. Use calls to the accessor methods to print what television was turned on.

7. Use calls to the mutator methods to change the channel to the user's preference and decrease the volume by two.

8. Use calls to the accessor methods to print the changed state of the portable.

9. Compile and debug this class.

10. Run TelevisionDemo again.

11. The output for task #5 will appear after the output from above, since we added onto the bottom of the program. The output for task #5 is shown below.

OUTPUT (boldface is user input)
A 19 inch Sharp has been turned
on. What channel do you want?
7
Channel: 7 Volume: 18