Module: Core Java

Session 3: Classes, Objects and Constructors Practice

* This is a practice session; you will work on classes, objects and constructors assignments.
* You can discuss your doubts with the trainer.

**Assignments:**

**Assignment 1 - Working with Object-Oriented Principles Assignment**

The goal of this assignment is to become familiar with object-oriented concepts, including classes, methods, and attributes. You also learn how to read a UML class model.

A description of the business rules contains information to enable you to determine the classes, methods, attributes, and associations that make up objects in the system. At the end of the assignment, you will have created a design model that includes the major classes in the business. The aim is to create a simple UML class model to show these objects.

Note that a model is a representation of something, and is subject to interpretation. Remember that no model is basically wrong—but some models are more usable than others.

Description of the Order Entry Area of the Business

The Order Entry component of the business is now becoming automated. In most respects, the process of ordering products is rather simple. The customers select the items that they want from a list of products. Your customers must be included in your system.You keep information about your customers such as name, address, and telephone number. You also assign a unique customer ID to each new customer. For customers that are companies, you track a contact person and provide for a discount on company purchases. You identify individual customers by their license number.

The order is not very complicated. Each order has a unique number that you can use to keep track of it and has information such as the customer who is responsible for the order, the order date, shipping mode (air or ground), and an order status. Each order can have multiple line items. You currently limit your customers to 10 items per order. Each item on an order has the product being purchased, the price, quantity, and the product category. A product category can be a composite category consisting of additional categories, or a leaf category.

You track many things about your products, and the key things include the name, description, and list price. Additionally, you want to include a warranty period, the supplier who distributes the product, a catalog URL to reference it on the Web, and a weight classification that is used when you calculate shipping costs. It is important for you to also track information about the products that are available, and where they are located. You have many warehouses to hold all your products.

Understanding Object-Oriented Principles

1. Identifying Business Classes, Attributes, and Methods

2. Identify some of the classes in the Order Entry business, limiting yourself to three. The process of identifying a class is to look for nouns that classify a group of things from the business description. Some nouns will describe the attributes of a class. Write a simple sentence or two describing each class to make sure that it is of interest to the business.

3. Identify a few attributes for each of the new classes. Remember that attributes may be other classes.

4. Define some behaviors (methods or operations) for each of the classes that you have discovered.

5. Look for classes that can inherit structure (attributes) and behavior (methods) from other classes. Modify your definitions to reflect the inheritance model.

**Assignment 2 - Class and Object basics Assignment**

The purpose of this assignment is to create a class and a separate program that uses that class by instantiating objects.

TalentSprint needs a program to calculate how much to pay their hourly employees. The US Department of Labor requires that employees get paid time and a half for any hours over 40 that they work in a single week. For example, if an employee works 45 hours, they get 5 hours of overtime, at 1.5 times their base pay. The State of Massachusetts requires that hourly employees be paid at least $8.00 an hour. TalentSprint requires that an employee not work more than 60 hours in a week.

An employee gets paid (hours worked) × (base pay), for each hour up to 40 hours.

For every hour over 40, they get overtime = (base pay) × 1.5.

The base pay must not be less than the minimum wage ($8.00 an hour). If it is, print an error.

If the number of hours is greater than 60, print an error message.

Using Methods, Classes, and Objects Assignment

Create a new class called TalentSprint. Write a method that takes the base pay and hours worked as parameters, and prints the total pay or an error. Write a main

method that calls this method for each of these employees:

|  |  |  |
| --- | --- | --- |
| Employee Name | Base Pay | Hours Worked |
| Employee 1 | $7.50 | 35 |
| Employee 2 | $8.20 | 47 |
| Employee 3 | $10.00 | 73 |

**Assignment 3 - Introduction to Classes**

## Objectives

* Be able to declare a new class
* Be able to write a constructor
* Be able to write instance methods that return a value
* Be able to write instance 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

## Introduction

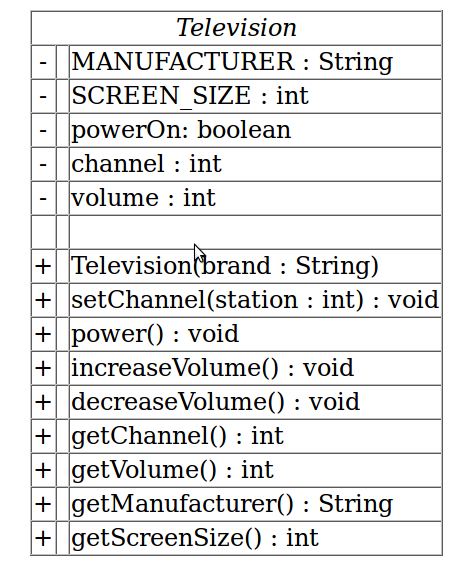
Everyone is familiar with a television. It is the object we are going to create in this assignment. First we need a blueprint. All manufacturers have the same basic elements in the televisions they produce as well as many options. We are going to work with a few basic elements that are common to all televisions. Think about a television in general. 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:

1. **manufacturer**  
   The manufacturer attribute will hold the brand name. This cannot change once the television is created, so will be a named constant.
2. **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.
3. **powerOn**  
   The powerOn attribute will hold the value true if the power is on, and false if the power is off.
4. **channel**  
   The channel attribute will hold the value of the station that the tele- vision is showing.
5. **volume**  
   The volume attribute will hold a number value representing the loud- ness (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.

1. **Set Channel**  
   The set Channel method will store the desired station in the channel field.
2. **power**  
   The power method will toggle the power between on and off, changing the value stored in the power On field from true to false or from false to true.
3. **Increase Volume**  
   The increase Volume method will increase the value stored in the volume field by 1.
4. **Decrease Volume**  
   The decrease Volume method will decrease the value stored in the volume field by 1.
5. **Get Channel**  
   The get Channel method will return the value stored in the channel field.
6. **Get Volume**  
   The get Volume method will return the value stored in the volume field.
7. **Get Manufacturer**  
   The get Manufacturer method will return the constant value stored in the *MANUFACTURER* field.
8. **Get Screen Size**  
   The get Screen Size 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 Modeling Language) diagram for this class as shown below.



## Task 1: Creating a new Class

1. In a new file, create a class definition called *Television*
2. That is, the file should be named *Television.java*
3. Put a program header (comments/documentation) at the top of the file // The purpose of this class is to model a television // Your name and today’s date
4. Declare the 2 constant fields listed in the UML diagram.
5. Declare the 3 remaining fields listed in the UML diagram.
6. Write a comment for each field indicating what it represents.
7. Save this file as Television.java.
8. Compile and debug. Do not run.

## Task 2: Writing a Constructor

* Create a constructor definition that has two parameters, a manufacturer’s brand and a screen size. These parameters will bring in information
* Inside the constructor, assign the values taken in from the parameters to the corresponding fields.
* Initialize the power On field to false (power is off), the volume to 20, and the channel to 2.
* Write comments describing the purpose of the constructor above the method header.
* Compile and debug. Do not run.

## Task 3: Methods

1. Define accessor methods called getVolume, getChannel, getManufacturer, and getScreenSize that return the value of the cor- responding 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 (remem- ber assignment statements evaluate the right hand side first, then assign the result to the left hand side variable.

1. 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.
2. Write javadoc comments above each method header.
3. Compile and debug. Do not run.

## Task 4: Running the application

* 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. Copy following code and save as TelevisionDemo.java in the same directory as Television.java.

import java.util.Scanner;

/\*\* This class demonstrates the Television class\*/

public class TelevisionDemo

{

public static void main(String[] args)

{

//create a Scanner object to read from the keyboard

Scanner keyboard = new Scanner (System.in);

//declare variables

int station; //the user's channel choice

//declare and instantiate a television object

Television bigScreen = new Television("Toshiba", 55);

//turn the power on

bigScreen.power();

//display the state of the television

System.out.println("A " + bigScreen.getScreenSize() + " inch " +

bigScreen.getManufacturer() + " has been turned on.");

//prompt the user for input and store into station

System.out.print("What channel do you want? ");

station = keyboard.nextInt();

//change the channel on the television

bigScreen.setChannel(station);

//increase the volume of the television

bigScreen.increaseVolume();

//display the the current channel and volume of the television

System.out.println("Channel: " + bigScreen.getChannel() +

" Volume: " + bigScreen.getVolume());

System.out.println("Too loud!! I am lowering the volume.");

//decrease the volume of the television

bigScreen.decreaseVolume();

bigScreen.decreaseVolume();

bigScreen.decreaseVolume();

bigScreen.decreaseVolume();

bigScreen.decreaseVolume();

bigScreen.decreaseVolume();

//display the current channel and volume of the television

System.out.println("Channel: " + bigScreen.getChannel() +

" Volume: " + bigScreen.getVolume());

System.out.println(); //for a blank line

//HERE IS WHERE YOU DO TASK 5

}

}

* Compile and run TelevisionDemo:

% javac TelevisionDemo.java

% java TelevisionDemo

and follow the prompts.

* If your output matches the output below, Television.java is complete and correct.

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. Declare another Television object called portable.
3. Instantiate portable to be a Sharp 19 inch television.
4. Use a call to the power method to turn the power on.
5. Use calls to the accessor methods to print what television was turned on.
6. Use calls to the mutator methods to change the channel to the user’s prefer- ence and decrease the volume by two.
7. Use calls to the accessor methods to print the changed state of the portable.
8. Compile and debug this class.
9. Run TelevisionDemo again.
10. 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.

A 19 inch Sharp has been turned on.

What channel do you want? 7

Channel: 7 Volume: 18

## At the end of assignment - Phrases you should now understand:

classes, object, instance variable, local variable, scope, (default) constructor

**Answer each of the questions below.**

1. What is an *instance* variable?
2. What is a *accessor* method?
3. What is a *mutator* method?
4. What is a *constructor*?
5. How is a *constructor* named?
6. What is a *constructor’s* return type?
7. What is a *stale data* item?
8. What is a method’s *signature*?