King Saud University
College of Computer & Information Science
CSC111 – Lab07
Objects II
All Sections

Objectives:

The students should learn how to:

1. Describe objects and classes, and use classes to model objects.
2. Use UML graphical notation to describe classes and objects.
3. Demonstrate how to define classes and create objects.
4. Create objects using default constructors.
5. Access objects via object reference variables.
6. Define a reference variable using a reference type.
7. Access an object's data and methods using the object member access operator (.).
8. Understand how to create and use methods, parameters and return values.

Submission Instructions:

1. ***Due date: Saturday, November 6th, 2021 at 11:59 pm***
2. You can discuss answers with your colleagues. **But no copying.**
3. Submit it to lms.ksu.edu.sa. **Email submissions will not be accepted**.
4. All classes in one java project. The project name must be: ***Lab07_ID_FirstName_LastName.zip***. For example: Lab07_123456789_Marwan_Almaymoni.zip
5. Use the default package.
6. Write your name and university ID as a comment at the start of all java files.

Introduction

In this lab, we will define a new class and call it **TV** and we will initialize three properties for this class. As we go, we will keep adding new methods to the TV class.

Part1

Write a class and call it **TV** with two **integer** variables **`channel`** and **`volumeLevel`**, and one **boolean** variable **`tvIsOn`**. Write the variables by completing the following pseudo-code:

```
public class TV {
	// data members
	// define instance variables channel, volumeLevel, tvIsOn
	/* modifier datatype variable name */
	/* modifier datatype variable name */
	/* modifier datatype variable name */
}
```

Part 2

In the previous part, you have defined a new class with instance variables to store the channel, the volume level and the status of the TV. In this part, you will use methods to modify the status of the TV (i.e., instance variable **`tvIsOn`**) rather than modifying the instance variable directly.

Add to the TV class you defined in part 1 two methods for modifying the instance variable on which are **`turnTvOn()`** and **`turnTvOff()`**. When you want to turn the TV on you call **`turnTvOn()`** (which will set on to **`true`**) and **`turnTvOff()`** for turning the TV off (which will set on to **`false`**).

(Note that both methods do NOT return any value and do NOT have any parameter)

```
public class TV {
	// data members
	// define instance variables channel, volumeLevel, tvIsOn
```

```
	/* modifier datatype variable name */
	/* modifier datatype variable name */
	/* modifier datatype variable name */

	//turn on the tv
	public void turnTvOn() {
		tvIsOn = true;
	}
	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false
		/* */
	}
}
```

Part 3

Now write a method **`isTvOn()`** to check whether the TV is on or off (i.e. return the value of the variable **`tvIsOn`**). See the example below:

```
public class TV {
	// data members
	// define instance variables channel, volumeLevel, tvIsOn
	/* modifier datatype variable name */
	/* modifier datatype variable name */
	/* modifier datatype variable name */

	//turn on the tv
	public void turnTvOn() {
		tvIsOn = true;
	}
	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false
		/* */
	}
	// returns true of tv is turned on
	/*modifier */ /* returntype */  isTvOn(){
		//return value of on
		/*    */
	}
}
```

Part 4

Now add two more methods to modify volume level. Each of the two methods takes an **integer** as a parameter. **`volumeLevelUp(int vol)`** and **`volumeLevelDown(int vol)`** should raise or lower the volume level by **`vol`** number of levels. The volume level must be between 0 and 8. If **`vol`** is negative, don't apply it. For example:

- if **`volumeLevel`** = 2, calling **`volumeLevelUp(3)`** makes **`volumeLevel`** be 2 + 3 = 5.
- if **`volumeLevel`** = 7, calling **`volumeLevelUp(4)`** makes **`volumeLevel`** be 7 + 4 = 11 ⇒ 8.
- if **`volumeLevel`** = 3, calling **`volumeLevelUp(-2)`** doesn't change **`volumeLevel.`**
- if **`volumeLevel`** = 2, calling **`volumeLevelDown(3)`** makes **`volumeLevel`** be 2 - 3 = -1 ⇒ 0.
- if **`volumeLevel`** = 6, calling **`volumeLevelDown(5)`** makes **`volumeLevel`** be 6 - 5 = 1.
- if **`volumeLevel`** = 4, calling **`volumeLevelDown(-7)`** doesn't change **`volumeLevel.`**

```
public class TV {
	// data members
	// define instance variables channel, volumeLevel, tvIsOn
	/* modifier datatype variable name */
	/* modifier datatype variable name */
	/* modifier datatype variable name */

	//turn on the tv
	public void turnTvOn() {
		tvIsOn = true;
	}
	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false
		/* */
	}
	// returns true of tv is turned on
	/*modifier */ /* returntype */  isTvOn(){
		//return value of on
		/*    */
	}

	// raises volume up such that it does not exceed 8
	public void volumeLevelUp(int vol) {
		if (/* value of vol is not negative */){
			/* calculate new volume level */
			if (/* value of new volume level is more than maximum */)
				/* set volumeLevel value to maximum */
		}
	}
```

```
	// lowers volume down such that it does not go below 0
	/*modifier */ /* returntype */ /* method name */(/*parameters*/) {
		//method body similar to volumeLevelUp
		/* */
	}
}
```

Part 5

Now add two more methods that take an **integer** as a parameter and modify the channel using the value of that integer. **`channelUp(int ch)`** and **`channelDown(int ch)`** should increment or decrement **`channel`** by **`ch`** number of channels. The channel must be between 0 and 99. If it goes beyond, they loop around. For example:

- if **`channel`** = 2, calling **`channelUp(3)`** makes **`channel`** be 2 + 3 = 5.
- if **`channel`** = 98, calling **`channelUp(4)`** makes **`channel`** be 98 + 4 = 102 ⇒ 2.
- if **`channel`** = 3, calling **`channelUp(-2)`** doesn't change **`channel.`**
- if **`channel`** = 2, calling **`channelDown(3)`** makes **`channel`** be 2 - 3 = -1 ⇒ 99.
- if **`channel`** = 6, calling **`channelDown(5)`** makes **`channel`** be 6 - 5 = 1.
- if **`channel`** = 4, calling **`channelDown(-7)`** doesn't change **`channel.`**

```
public class TV {
	// data members
	// define instance variables channel, volumeLevel, tvIsOn
	/* modifier datatype variable name */
	/* modifier datatype variable name */
	/* modifier datatype variable name */

	//turn on the tv
	public void turnTvOn() {
		tvIsOn = true;
	}
	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false
		/* */
	}
	// returns true of tv is turned on
	/*modifier */ /* returntype */  isTvOn(){
		//return value of on
		/*    */
	}
```

```
	// raises volume up such that it does not exceed 8
	public void volumeLevelUp(int vol) {
		if (/* value of vol is not negative */){
			/* calculate new volume level */
			if (/* value of new volume level is more than maximum */)
				/* set volumeLevel value to maximum */
		}
	}
	// lowers volume down such that it does not go below 0
	/*modifier */ /* returntype */ /* method name */(/*parameters*/) {
		//method body similar to volumeLevelUp
		/* */
	}

	// channelUp method goes here
	/*    */

	// channelDown method goes here
	/*    */
}
```

Part 6

Add a new method **`toString()`** that returns all current information of the TV itself. It should **return** the TV info in a format similar to the following example:

If TV is on:

```
TV is on and the current channel is 5 and the current volume level is 8.
```

If TV is off:

```
TV is off.
```

Part 7

Test your program by creating a new class **TestTV** with a **`main`** method. Then do the following (follow every step by printing the TV info using method **`toString()`**):

1. Create a TV object.
2. Check if the TV is off and if turn the tv on.

3. Go to channel 20. Then go to channel 30.
4. Raise the volume by 4. Then lower it by 3.
5. Turn the TV off.

Sample Run

```
TV is off.
TV is on and the current channel is 0 and the current volume level is 0.
TV is on and the current channel is 20 and the current volume level is 0.
TV is on and the current channel is 30 and the current volume level is 0.
TV is on and the current channel is 30 and the current volume level is 4.
TV is on and the current channel is 30 and the current volume level is 1.
TV is off.
```

Class **TestTV** pseudo-code:

```
public class TestTV {
	public static void main(String[] args) {
		/* create the object tv1 */
		if (/* check if tv is off */)
			/* turn the object tv1 on */
		System.out.println(tv1.toString());
		System.out.println(/*get tv info using toString() */);
		/* go to channel 20, then print toString() */
		/* go to channel 30, then print toString() */
		/* raise volume by 4, then print toString() */
		/* lower volume by 3, then print toString() */
		/* turn the TV Off, then print toString() */
	}
}
```

Part 8

Draw the UML for class **TV** (Hint: see UML for class LinearEquation in next exercise).

TV

Lab Exercise 2 (Optional)

Design a class named **LinearEquation** for a 2 x 2 system of linear equations:

$$ax + by = e$$

$$cx + dy = f$$

A system of linear equations can be solved using Cramer's rule as following:

$$x = \frac{ed-bf}{ad-bc} \qquad y = \frac{af-ec}{ad-bc}$$

The class contains:

- Data fields **a, b, c, d, e**, and **f**.
- A method named **`isSolvable()`** that returns true if ad - bc is not 0.
- Methods **`solveX()`** and **`solveY()`** that return the solution for the equation.

Draw the UML diagram for the class and then implement the class. Write a test program that prompts the user to enter **a, b, c, d, e**, and **f** and displays the solution. If **ad - bc** is **0**, report that **"The system has no solution."**

Sample Runs

```
Enter a, b, c, d, e, f: 9 4 3 -5 -6 -21 ↵
x is -2.0 and y is 3.0
```

```
Enter a, b, c, d, e, f: 1 2 2 4 5 5 ↵
The system has no solution
```

UML

Unlike the previous program, in this program we are going to solve everything at once, i.e., write the whole class at once. First phase is to design your program as an OOP

program. Draw UML diagrams for the two classes, **LinearEquation** and **TestLinearEquation**.

LinearEquation
+ a: double + b: double + c: double + d: double + e: double + f: double
+ isSolvable(): boolean + solveX(): double + solveY(): double

TestLinearEquation
+ main(): void

Partial Solution

Now write your program. Construct two classes **LinearEquation** and **TestLinearEquation** in the same file.

```
import java.util.Scanner;
public class TestLinearEquation {
	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		/* create an object of type equation */

		System.out.print("Enter a, b, c, d, e, f: ");
		equation.a = input.nextDouble();
		//read remaining fields b, c, d, e, f

		if (/*check if equation is solvable */) {
			System.out.println("x is " + /* call method solveX */
```

```
                        + " and y is " + /*call method solveY */  );
            } else {
                  System.out.println("The system has no solution");
                  System.exit(0);
            }
      }
}

class LinearEquation {
      //data members

      public /* return type */ isSolvable() {
            return /* boolean expression to check if solvable */;
      }

      /*modifier*/ /*return type*/ solveX() {
            double x = /* calculate the solution */
            /* return the solution */
      }

      //write method solveY()
      /*   */
}
```