**king Saud University**
**College of Computer &Information Science**
**CSC111 – Lab**
**Object – II –**
**All Sections**

---

## Objectives:

- To describe objects and classes, and use classes to model objects.
- To use UML graphical notation to describe classes and objects.
- To demonstrate how to define classes and create objects.
- To create objects using default constructors.
- To access objects via object reference variables.
- To define a reference variable using a reference type.
- To access an object's data and methods using the object member access operator (.).

## Lab Exercise 1

### 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 such as `turnOn()` and `turnOff()` methods for tuning the TV off and on.

## Part 1

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

```
public class TV
{
   // data members
   // define instance variables channel, volumeLevel,
   on
   /* 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 a variable to store the channel , a variable to store the volume level and a variable to store status of the TV on. In this part you will use methods to modify the variable `on` rather than modifying the variable directly .

Add to the TV class you defined in part 1 two methods for modifying the variables which are `turnOn()` and `turnOff()`. When you want to turn the TV on you call `turnOn()` (i.e. you set `on` to true) and `turnOff()` for turning the TV off (i.e. you 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, on
     /* modifier  datatype  variable name*/
     /* modifier  datatype  variable name*/
     /* modifier  datatype  variable name*/

     //turn on the tv
     public void turnOn() {
          on = true;
     }

     //turn off the tv
     public /* returntype */ /* method name */() {
          //set on to false

     }

}
```

## Part 3

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

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

	//turn on the tv
	public void turnOn() {
			on = true;
	}

	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false

	}

	// returns true of tv is turned on
	/*modifier */ /* returntype */  isOn(){
		//return value of on
	}

}
```

## Part 4

Now add two more methods that take an integer as a parameter and modify volume level by that integer. `volumeLevelUp(int vol)` and `volumeLevelDown(int vol)` should raise or lower the volume level by vol number of levels. For example if `volumeLevel` is 5 and you called `volumeLevelUp(3)` the new `volumeLevel` should be equal to 5 + 3 = 8. The volume level must be between 0 and 8.

```
public class TV
```

```
{
	// data members
	// define instance variables channel, volumeLevel, on
	/* modifier  datatype  variable name*/
	/* modifier  datatype  variable name*/
	/* modifier  datatype  variable name*/

	//turn on the tv
	public void turnOn() {
			on = true;
	}


	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false

	}

	// returns true of tv is turned on
	/*modifier */ /* returntype */  isOn(){
		//return value of on
	}

	// raises volume up such that it does not exceed 8
	public void volumeLevelUp(int vol) {
		/* calculate new volume level */
		if (/* value of new volume level less than 8 */){
			/* set volumeLevel value to new volume level */
		}
	}
	// 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 volume level by that integer. `channelUp(int ch)` and `channelDown(int ch)` should increment or decrement the channel by ch number of channels. For example if `channel` is 5 and you called `channelUp(10)` the new `channel` value should be equal to 5 + 10 = 15. The channel must be between 0 and 100.

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

	//turn on the tv
	public void turnOn() {
			on = true;
	}


	//turn off the tv
	public /* returntype */ /* method name */() {
		//set on to false

	}

	// returns true of tv is turned on
	/*modifier */ /* returntype */  isOn(){
		//return value of on
	}

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

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

## Part 7

Test your program by creating a new class `TestTV` with a `main` method. In this class you do the following:

1- Create a TV object.
2- Turn the tv on.
3- Go to channel 20 (java always initializes your int data members to 0, so the value of the channel when you create the TV object is 0).
4- Raise the volume by 4.
5- Check if the TV is on and if so print its information using method `toString()`.
6- Turn the TV off.

### Sample Run

```
TV is on and current channel is 20 and current
volume level is 4.
```

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

## 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:

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

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 Run 1

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

## Sample Run 2

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

## UML

Unlike 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<br>b: double<br>c: double<br>d: double<br>e: double<br>f: double |
| isSolvable(): boolean<br>solveX(): double<br>solveY(): double |

| TestLinearEquation |
| --- |
| |
| main(): void |

## Solution

Now write your program. Construct two classes LinearEquation and TestLinearEquation.

```java
import java.util.Scanner;

public class TestLinearEquation {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    /* create object 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()


}
```