PROGRAM 1

Write a class Triangle, which has two member variables base of type int, and height of type int.

Write a constructor which initialises the base and the height of a Triangle instance.

Write a method getArea() that returns the area of the Triangle as a double.

Write a method show(), to print the dimensions and area of the Triangle instance.

Write a method compare(Triangle t1, Triangle t2), which determines compares the area of two given Triangle objects (hint: recall the Float class compare() method used in Lab #2).

In the main method of the Triangle class, obtain user input for the Triangle's base and height.

If the user wishes to do a comparison, ask for the dimensions of Triangle t1 and Triangle t2.

SOURCE CODE

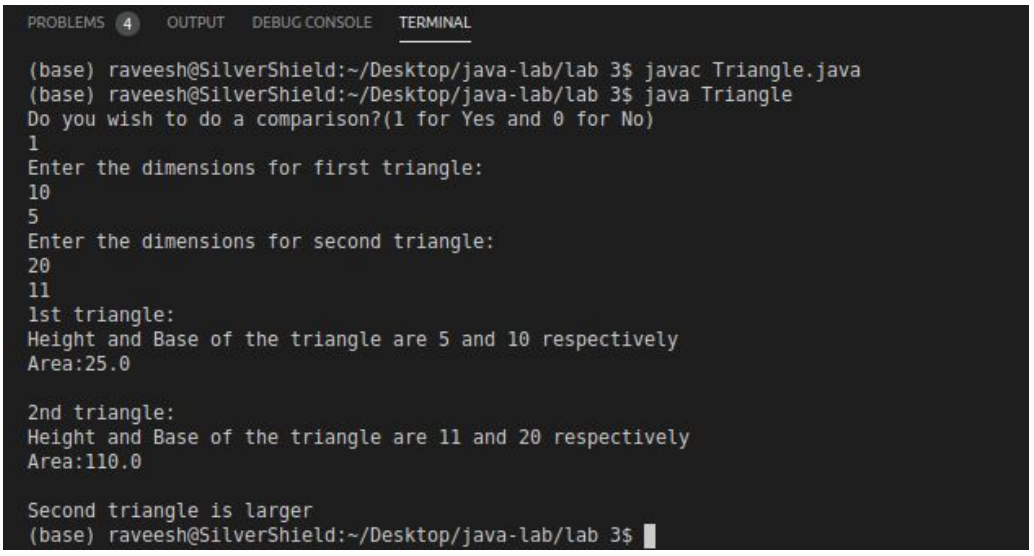
```
import java.util.*;


class Triangle {
   int base;
   int height;

   Triangle() {
      base = 0;
      height = 0;
   }

   Triangle(int b, int h) {
      base = b;
      height = h;
   }
```

```
  double getArea() {
    return 0.5 * this.base * this.height;
  }

  void show() {
    System.out
        .println("Height and Base of the triangle are " + this.height + " and " + this.base + "
respectively");
    System.out.println("Area:" + this.getArea());
  }

  static void compare(Triangle t1, Triangle t2) {
    if (t1.getArea() == t2.getArea())
      System.out.println("Triangles are equal");
    else if (t1.getArea() > t2.getArea())
      System.out.println("First triangle is larger");
    else
      System.out.println("Second triangle is larger");
  }

  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    int b, h;

    System.out.println("Do you wish to do a comparison?(1 for Yes and 0 for No)");
    int ch = sc.nextInt();
```

```
switch (ch) {
   case 0:
      System.out.println("Enter the dimensions for Triangle:");
      b = sc.nextInt();
      h = sc.nextInt();
      checkValidity(b, h);
      Triangle t = new Triangle(b, h);

      t.show();

      break;
   case 1:
      System.out.println("Enter the dimensions for first triangle:");
      b = sc.nextInt();
      h = sc.nextInt();
      checkValidity(b, h);
      Triangle t1 = new Triangle(b, h);

      System.out.println("Enter the dimensions for second triangle:");
      b = sc.nextInt();
      h = sc.nextInt();
      checkValidity(b, h);
      Triangle t2 = new Triangle(b, h);

      System.out.println("1st triangle:");
      t1.show();
```

```
            System.out.println();

            System.out.println("2nd triangle:");
            t2.show();
            System.out.println();

            compare(t1, t2);

            break;

         default:
            System.out.println("Invalid Input");
      }
      scanner.close();
   }

   public static void checkValidity(int base, int height) {
      if (base <= 0 || height <= 0) {
         System.out.println("Base/Height cannot be 0 or negative");
         System.exit(0);
      }
   }

}
```

OUTPUT



PROGRAM 2

Implement the Equipment class from the IFCS, according to the following class diagram:

Equipment

- id : String

- description: String

+ Equipment(id:String, desc:String)

+ getId() : String

+ getDesc() : String

Write an IFCSManager class to maintain an array of Equipment objects, sorted according to Equipment id. (Hint: refer to Lab #2 Question 2).

The IFCSManager will

add new Equipment instances

remove an Equipment instance specified by its id

given an id, report if the Equipment instance resides in the Lab

display the list of Equipment instances in the Lab.

SOURCE CODE

```
/**
     * The IFCSManager manages a list of Equipment
     *
     * @author mahak makharia
     */
    import java.util.*;

    public class IFCSManager {

      private Equipment[] eqpList;
      private int length;

      public static void main(String[] args) {

        boolean quit = false;
        Scanner sc = new Scanner(System.in);
        IFCSManager myMgr = new IFCSManager();

        while (!quit) {
          System.out.println("Enter choice number:");
          System.out.println("1. Insert\n2. Remove\n3. Report\n4. Display\n5. Quit");
          int choice = sc.nextInt();
          switch (choice) {
```

```
case 1: {
   System.out.print("Enter id:");
   String id = sc.next();
   System.out.print("Enter description:");
   String desc = sc.next();
   if (myMgr.insert(new Equipment(id, desc)))
      System.out.println("Equipment Added");
   else
      System.out.println("Equipment can't be added");
}
   break;
case 2: {
   System.out.print("Enter id:");
   String id = sc.next();
   if (myMgr.remove(id))
      System.out.println("Equipment Removed");
   else
      System.out.println("Equipment with " + id + " can't be found.");
}
   break;
case 3: {
   System.out.print("Enter id:");
   String id = sc.next();
   if (myMgr.find(id))
      System.out.println("Equipment Available");
   else
```

```
                System.out.println("Equipment not in Lab");
            }
               break;
            case 4: {
               myMgr.display();
            }
               break;
            case 5: {
               System.out.println("Exiting...");
               quit = true;
            }
               break;
            default:
               System.out.println("Invalid Choice");
         }
      }
      scanner.close();

   }

   /**
    * Default constructor, initialises Equipment list
    */
   public IFCSManager() {
      eqpList = new Equipment[10];
      length = 0;
```

```
}

/**
 * Inserts the Equipment into the list
 *
 * @param eqp the Equipment instance to be inserted into the list
 * @return true if Equipment is successfully inserted
 */
public boolean insert(Equipment eqp) {
   if (eqp.id.equals("null")) {
      System.out.println("Error - id cannot be null");
      return false;
   }
   if (eqp.description.equals("null")) {
      System.out.println("Error - description cannot be null");
      return false;
   }
   if (length + 1 < 11) {
      eqpList[length] = eqp;
      length++;
      return true;
   }
   return false;

}
```

```
/**
 * Removes the Equipment instance, specified by its id, from the list
 *
 * @param id
 * @return true if Equipment is successfully removed
 */
public boolean remove(String id) {
   if (id.equals("null")) {
      System.out.println("Error - id cannot be null");
      return false;
   }
   for (int i = 0; i < length; i++) {
      if (eqpList[i].id.equals(id)) {
         for (int j = i + 1; j < length; j++) {
            eqpList[j - 1] = eqpList[i];
         }
         length--;
         return true;
      }
   }
   return false;

}

/**
 * Locates the Equipment instance with the specified id
```

```
     *
     * @param id
     * @return
     */
    public boolean find(String id) {
        if (id.equals("null")) {
            System.out.println("Error - id cannot be null.");
            return false;
        }
        for (int i = 0; i < length; i++)
            if (eqpList[i].id.equals(id))
                return true;
        return false;

    }

    public void display() {
        for (int i = 0; i < length; i++) {
            System.out.println("id=" + eqpList[i].id + ", " + "desc=" + eqpList[i].description);
        }
    }

}

class Equipment {
    String id, description;
```

```
    public Equipment(String id, String desc) {
        this.id = id;
        this.description = desc;
    }

    String getId() {
        return this.id;
    }

    String getDesc() {
        return this.description;
    }
}
```

```
(base) raveesh@SilverShield:~/Desktop/java-lab/lab 3$ java IFCSManager
Enter choice number:
1. Insert
2. Remove
3. Report
4. Display
5. Quit
1
Enter id:123456
Enter description:oscilloscope
Equipment Added
Enter choice number:
1. Insert
2. Remove
3. Report
4. Display
5. Quit
1
Enter id:null
Enter description:null
Error - id cannot be null
Equipment can't be added
Enter choice number:
1. Insert
2. Remove
3. Report
4. Display
5. Quit
1
Enter id:12345
Enter description:multimeter
Equipment Added
Enter choice number:
1. Insert
2. Remove
3. Report
4. Display
5. Quit
4
id=123456, desc=oscilloscope
id=12345, desc=multimeter
Enter choice number:
1. Insert
2. Remove
```


Name: Mahak Makharia, Section: 3-A, Roll. No.: 98, Group: C3