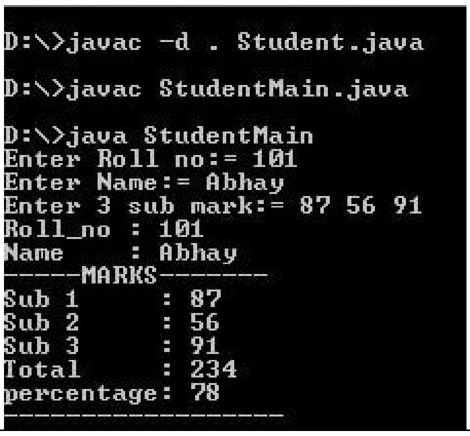
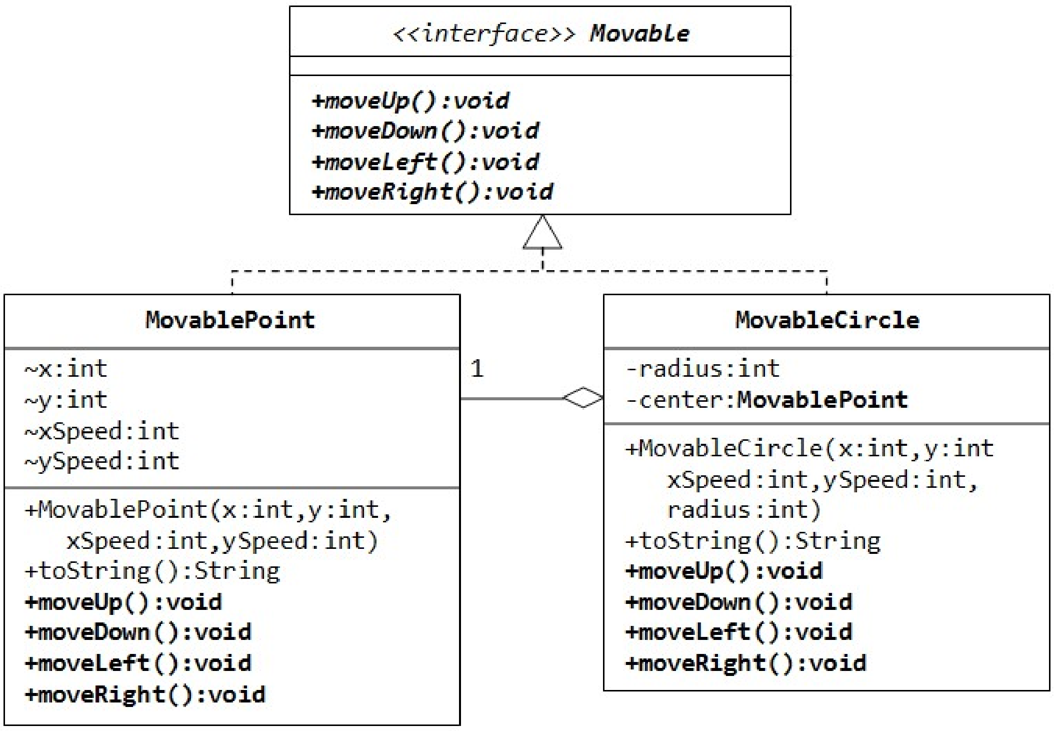
1.	Create a Package ***btech*** which has one class ***Student***. Accept student detail through parameterized constructor of ***Student*** class. Write a method ***display*** ()to display the student details. Create another class ***Test*** containing the main method which will use the package ***btech*** and calculate total marks and percentage of marks. One sample output is shown below. 
	```
	D:\>javac -d . Student.java

	D:\>javac StudentMain.java

	D:\>java StudentMain
	Enter Roll no:= 101
	Enter Name:= Abhay
	Enter 3 sub mark:= 87 56 91
	Roll_no : 101
	Name     : Abhay
	-----MARKS-------
	Sub 1     : 87
	Sub 2     : 56
	Sub 3     : 91
	Total     : 234
	percentage: 78
	------------------
	```

2.	Create a sub-package called ***arithmetic***under the package ***btech***. The ***arithmetic*** package should contain a class ***MyMath*** having methods to deal with different arithmetic operations (addition, subtraction, multiplication, division and mod). Create a class ***Test*** containing the main method which will use the methods of sub-package ***arithmetic***.
3.	Create a sub-package named ***shapes***under a package ***org***. Create some classes in the package representing some common geometric shapes like ***Square***, ***Triangle***, ***Circle*** and so on. The classes should contain the ***area*( )** and ***perimeter*( )** methods in them. Compile the package. Use this package to find area and perimeter of different shapes as chosen by the user.
4.	Run the programs under section 7.5 (Access Protection) present in page 82 of class note to test the visibility of class members in subclasses in the same package, non-subclasses in the same package, subclasses in different packages, classes that are neither in the same package nor subclasses. Uncomment the commented lines, test and analyse the output.

Assignment 8

Topic: Interfaces

Sl. No.	Question

1.	Suppose that we have a set of objects with some common behaviours: they could move up, down, left or right. The exact behaviours (such as how to move and how far to move) depend on the objects themselves. One common way to model these common behaviors is to define an *interface* called **Movable**, with abstract methods **moveUp()**, **moveDown()**, **moveLeft()**and **moveRight()**. The classes that implement the Movable interface will provide actual implementation to these abstract methods.
	Write two concrete classes - **MovablePoint** and **MovableCircle** - that implement the **Movable** interface.
	 <<interface>> Movable
	+moveUp():void
	+moveDown():void
	+moveLeft():void
	+moveRight():void
	MovablePoint
	~x:int
	~y:int
	~xSpeed:int
	~ySpeed:int
	+MovablePoint(x:int,y:int, xSpeed:int,ySpeed:int)
	+toString():String
	+moveUp():void
	+moveDown():void
	+moveLeft():void
	+moveRight():void
	1
	MovableCircle
	-radius:int
	-center:MovablePoint
	+MovableCircle(x:int,y:int xSpeed:int,ySpeed:int, radius:int)
	+toString():String
	+moveUp():void
	+moveDown():void
	+moveLeft():void
	+moveRight():void
	The code for the interface Movable is as follows:
	`public interface Movable {  // saved as "Movable.java"`
	`   public void moveUp();`
	`   ......`
	`}`
	For the **MovablePoint** class, declare the instance variable **x**, **y**, **xSpeed** and **ySpeed** with package access as shown with '~' in the class diagram (i.e., classes in the same package can access these variables directly). For the **MovableCircle** class, use a **MovablePoint** to represent its center (which contains four variable x, y, xSpeed and ySpeed). In other words, the **MovableCircle** composes a **MovablePoint**, and its radius.

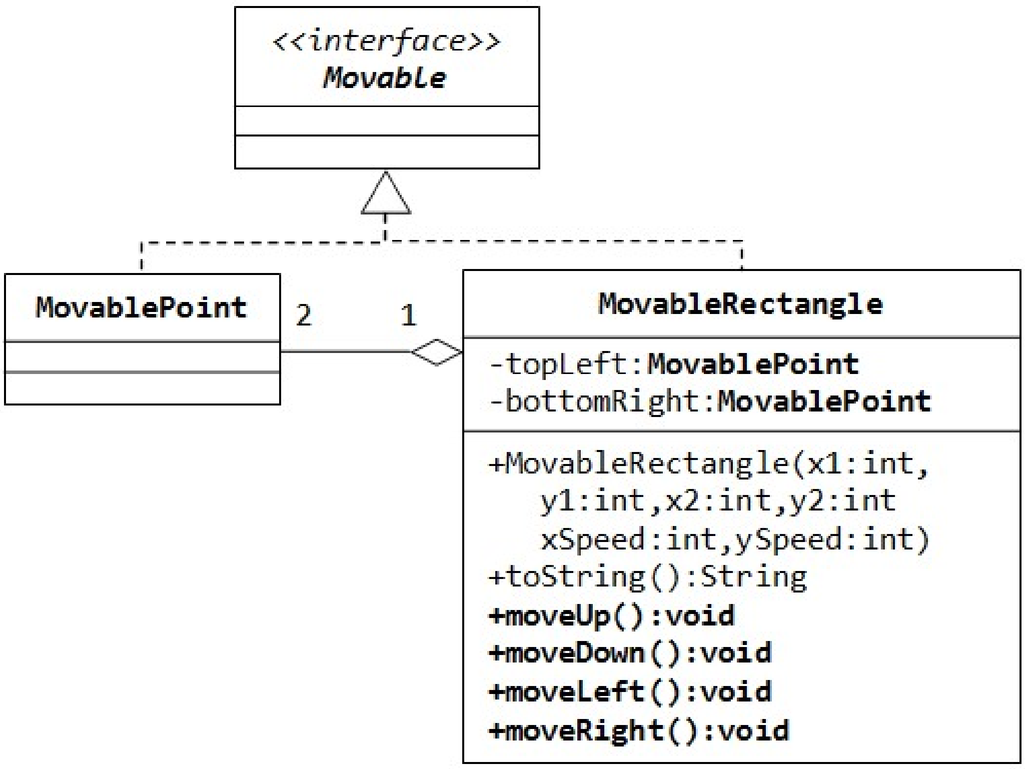
	See code below

```
public class MovablePoint implements Movable { // saved as "MovablePoint.java"
// instance variables
int x, y, xSpeed, ySpeed;     // package access

// Constructor
   public MovablePoint(int x, int y, intxSpeed, intySpeed) {
this.x = x;
      ......
   }
   ......

// Implement abstract methods declared in the interface Movable
   @Override
   public void moveUp() {
      y -= ySpeed;   // y-axis pointing down for 2D graphics
   }
   ......
}
public class MovableCircle implements Movable { // saved as "MovableCircle.java"
// instance variables
   private MovablePointcenter;   // can use center.x, center.y directly
                                  //  because they are package accessible
   private int radius;

// Constructor
   public MovableCircle(int x, int y, intxSpeed, intySpeed, int radius) {
// Call the MovablePoint's constructor to allocate the center instance.
center = new MovablePoint(x, y, xSpeed, ySpeed);
      ......
   }
   ......

// Implement abstract methods declared in the interface Movable
   @Override
   public void moveUp() {
center.y -= center.ySpeed;
   }
   ......
}
```

Write a test program and try out these statements:

```
Movable m1 = new MovablePoint(5, 6, 10, 15);     // upcast
System.out.println(m1);
m1.moveLeft();
System.out.println(m1);

Movable m2 = new MovableCircle(1, 2, 3, 4, 20);  // upcast
```

	`System.out.println(m2);`
	`m2.moveRight();`
	`System.out.println(m2);`

2.	Write a new class called **MovableRectangle**, which composes two MovablePoints (representing the top-left and bottom-right corners) and implementing the **Movable** Interface. Make sure that the two points has the same speed.  <<interface>> Movable MovablePoint 2 1 MovableRectangle -topLeft:MovablePoint -bottomRight:MovablePoint +MovableRectangle(x1:int, y1:int,x2:int,y2:int xSpeed:int,ySpeed:int) +toString():String +moveUp():void +moveDown():void +moveLeft():void +moveRight():void