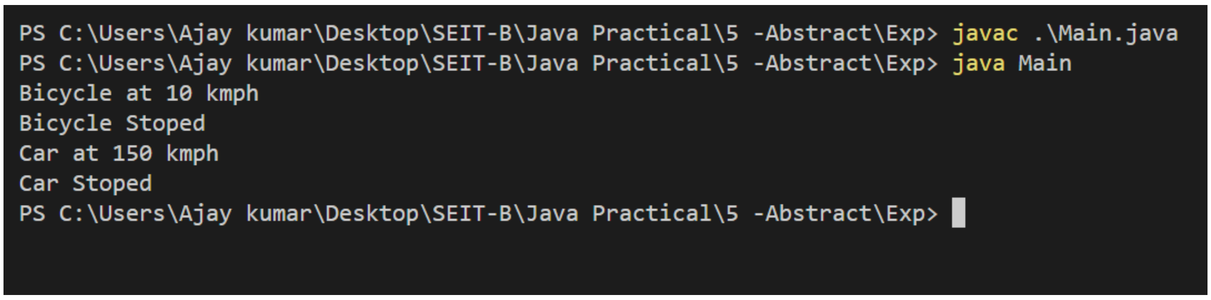
i. Create an interface vehicle and classes like bicycle, car, bike etc, having common functionalities and put all the common functionalities in the interface. Classes like Bicycle, Bike, car etc implement all these functionalities in their own class in their own way.

Code:

```
interface Vehicle {
    void speed();
    void declerate();
}
class Bicycle implements Vehicle {
    public void speed() {
        System.out.println("Bicycle at 10 kmph");
    }
    public void declerate() {
        System.out.println("Bicycle Stoped");
    }
}
class Car implements Vehicle {
    public void speed() {
        System.out.println("Car at 150 kmph");
    }
    public void declerate() {
        System.out.println("Car Stoped");
    }
}
class Motorcycle implements Vehicle {
    public void speed() {
        System.out.println("MotorCycle at 100 kmph");
    }
    public void declerate() {
        System.out.println("MotorCycle Stoped");
    }
}
class Truck implements Vehicle {
    public void speed() {
        System.out.println("Truck at 120 kmph");
    }
    public void declerate() {
        System.out.println("Truck Stoped");
    }
}

public class Main {
    public static void main(String args[]) {
        Bicycle b = new Bicycle();
        b.speed();
```

```java
        b.declerate();

        Car c = new Car();
        c.speed();
        c.declerate();
    }
}
```

Output:



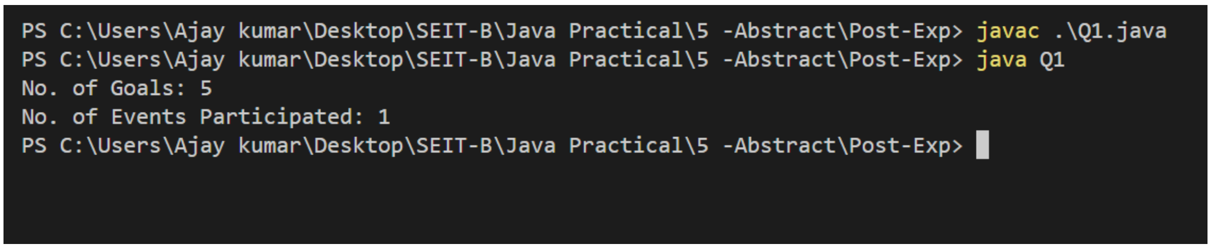
```
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Exp> javac .\Main.java
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Exp> java Main
Bicycle at 10 kmph
Bicycle Stoped
Car at 150 kmph
Car Stoped
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Exp>
```

Consider a hierarchy, where a sportsperson can either be an athlete or a hockey player. Every sportsperson has a unique name. An athlete is characterized by the event in which he/she participates; where as a hockey player is characterised by the number of goals scored by him/her.

Code:

```
class SportsPerson {
  String name;
}


class Athlete extends SportsPerson {
  int eventsParticipated;
  Athlete (String name) {
    this.name = name;
    eventsParticipated = 0;
  }
  Athlete (String name, int participated) {
    this.name = name;
    this.eventsParticipated = participated;
  }
  void participate() {
    this.eventsParticipated ++;
  }
}


class Hockey extends SportsPerson {
  int noOfGoals;
  Hockey (String name) {
    this.name = name;
    this.noOfGoals = 0;
  }
  Hockey (String name, int goals) {
    this.name = name;
    this.noOfGoals = goals;
  }
  void goal() {
    this.noOfGoals ++;
  }
}

class Q1 {
  public static void main(String[] args) {
    Hockey hockey = new Hockey("Ajaykumar", 4);
    hockey.goal();
    System.out.println("No. of Goals: "+hockey.noOfGoals);

    Athlete athlete = new Athlete("Kevin");
    athlete.participate();
    System.out.println("No. of Events Participated:
"+athlete.eventsParticipated);
  }
}
```

Output:



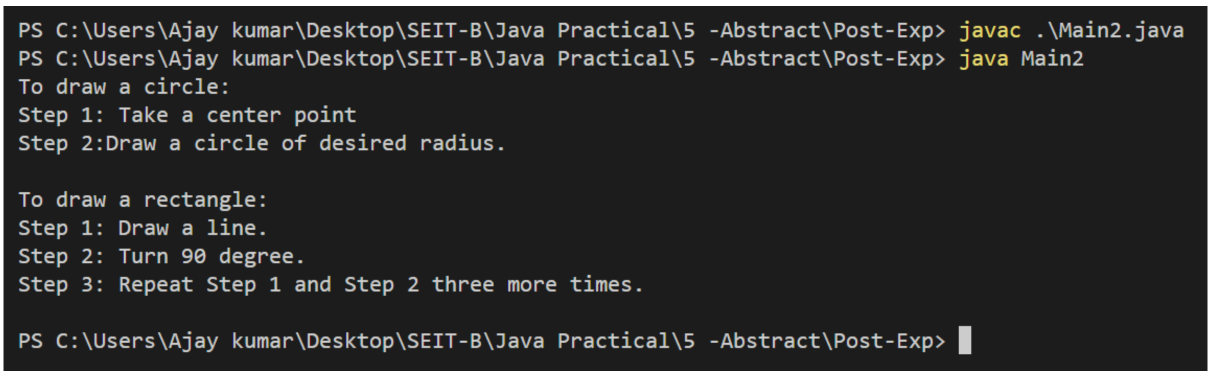
```
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Post-Exp> javac .\Q1.java
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Post-Exp> java Q1
No. of Goals: 5
No. of Events Participated: 1
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Post-Exp>
```

Create one abstract class shape and abstract method draw. Create class rectangle and circle that will inherit method from Shape class.Make necessary assumptions

Code:

```
abstract class Shape {
    abstract void draw();
}
class Rectangle extends Shape {
    void draw() {
        System.out.println("To draw a rectangle: ");
        System.out.println("Step 1: Draw a line.\nStep 2: Turn 90 
degree.\nStep 3: Repeat Step 1 and Step 2 three more times.\n");
    }
}
class Circle extends Shape {
    void draw() {
        System.out.println("To draw a circle: ");
        System.out.println("Step 1: Take a center point\nStep 2:Draw a 
circle of desired radius.\n ");
    }
}
class Main2 {
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.draw();

        Rectangle rectangle = new Rectangle();
        rectangle.draw();
    }
}
```

Output:



```
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Post-Exp> javac .\Main2.java
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Post-Exp> java Main2
To draw a circle:
Step 1: Take a center point
Step 2:Draw a circle of desired radius.

To draw a rectangle:
Step 1: Draw a line.
Step 2: Turn 90 degree.
Step 3: Repeat Step 1 and Step 2 three more times.

PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\5 -Abstract\Post-Exp>
```