ANSWER SHEET

What is Object Oriented Paradigm and Object Oriented Programming? How they are linked together.

ANSWER:

Object-Oriented Programming (OOP) is the term used to describe a programming approach based on objects and classes. The object-oriented paradigm allows us to organize software as a collection of objects that consist of both data and behavior. This is in contrast to conventional functional programming practice that only loosely connects data and behavior.

Explain polymorphism by programming example.

ANSWER:

Polymorphism:

The word polymorph is a combination of two words namely, ‘poly’ which means ‘many’ and ‘morph’ which means ‘forms’. Thus, polymorph refers to an object that can have many different forms.

Example:

```
class Area {
   private int len,wid;
   public void setLen(int len) {
      this.len = len;
   }
   public void setWid(int wid) {
      this.wid = wid;
   }
   public int CalculateArea() {
      return len * wid;
   }
   public void show(){
      System.out.println("Length Of Rectangle : "+len+"\nWidth of Rectangle : "+wid);
   }
}
```

ANSWER SHEET

```
class Circle extends Area {
   protected double PI = 3.14;
   public double CalculateArea(double Radius)
   {
      return PI * Radius * Radius;
   }
   @Override
   public void show() {
      super.show();
      System.out.println("Value of PI : "+PI);
   }
}
public class Main {
   public static void main(String[] args) {
            Circle cir = new Circle();
            cir.setLen(5);
            cir.setWid(7);
            cir.show();
      System.out.println("Area of Rectangle : "+cir.CalculateArea());
      System.out.println("Area of Circle : "+cir.CalculateArea(2.5));
   }
}
```

What is Inheritance in Object Oriented Programming, define its types with relevant code

ANSWER:

Inheritance:

ANSWER SHEET

The mechanism of basing an object or class upon another object or class, retaining similar implementation. Also defined as deriving new classes from existing ones and forming them into a hierarchy of classes.

Types of Inheritance:

1. **Single Inheritance:**

```
class Vehicle {
   public void accelerate(int speed){
      System.out.println("Vehicle accelerating at:"+ speed + "kmph");
   }
}
class FourWheeler extends Vehicle{
   public void showDetails(boolean powerSteer){
      if(powerSteer==true){
         System.out.println("Power Steering for four wheeler vehicle:Yes");
      }
      else{
         System.out.println("Power Steering for four wheeler vehicle:No");
      }
   }
}
public class Main {
   public static void main(String[] args) {
      FourWheeler fw = new FourWheeler();
      fw.accelerate(75);
      fw.showDetails(true);
   }
}
```

2. **Multi-level Inheritance:**

```
class Vehicle {
   public void accelerate(int speed){
      System.out.println("Vehicle accelerating at:"+ speed + "kmph");
   }
}
class FourWheeler extends Vehicle{
   public void showDetails(boolean powerSteer){
      if(powerSteer==true){
```

ANSWER SHEET

```
            System.out.println("Power Steering for four wheeler vehicle:Yes");
        }
        else{
            System.out.println("Power Steering for four wheeler vehicle:No");
        }
    }
}
class Car extends FourWheeler{
    public void color(String color){
        System.out.println("The color of car is : "+color);
    }
}
public class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.accelerate(95);
        c.showDetails(true);
        c.color("Dark Grey");
    }
}
```

3. Hierarchical Inheritance:

```
class Vehicle {
    public void accelerate(int speed){
        System.out.println("Vehicle accelerating at:"+ speed + "kmph");
    }
}
class FourWheeler extends Vehicle{
    public void showDetails(boolean powerSteer){
        if(powerSteer==true){
            System.out.println("Power Steering for four wheeler vehicle:Yes");
        }
        else{
            System.out.println("Power Steering for four wheeler vehicle:No");
        }
    }
}
class TwoWheeler extends Vehicle{
    public void showDetails(boolean powerSteer){
```

ANSWER SHEET

```
        if(powerSteer==true){
            System.out.println("Power Steering for Two wheeler vehicle:Yes");
        }
        else{
            System.out.println("Power Steering for two wheeler vehicle:No");
        }
    }
}
public class Main {
    public static void main(String[] args) {
        FourWheeler fw = new FourWheeler();
        fw.accelerate(75);
        fw.showDetails(true);

        TwoWheeler tw = new TwoWheeler();
        tw.accelerate(60);
        tw.showDetails(false);
    }
}
```

4. Multiple Inheritance (using interface)
5. Hybrid Inheritance (using interface)