

Text-7(set-1)

1. Create a class Employee with properties name and salary, and a constructor to initialize these properties. Create a subclass Manager that adds a property department and a constructor to initialize all properties. Demonstrate creating instances of both classes.

Program:

```
class Employee {  
    String name;  
    double salary;  
  
    Employee(String name, double salary) {  
        this.name = name;  
        this.salary = salary;  
    }  
    String getName() {  
        return name;  
    }  
  
    double getSalary() {  
        return salary;  
    }  
}  
  
class Manager extends Employee {  
    String department;  
  
    Manager(String name, double salary, String department) {  
        super(name,salary);  
        this.department = department;  
    }  
}
```

```

String getDepartment() {
    return department;
}

}

public class Main {
    public static void main(String[] args) {

        Employee emp = new Employee("polireddy", 500000.0);
        System.out.println("Employee Name: " + emp.getName());
        System.out.println("Employee Salary: " + emp.getSalary());

        Manager mgr = new Manager("saikumar", 80000.0, "microsoft company");
        System.out.println("\nManager Name: " + mgr.getName());
        System.out.println("Manager Salary: " + mgr.getSalary());
        System.out.println("Manager Department: " + mgr.getDepartment());
    }
}

```

Output:

```

java -cp /tmp/3Xgny36SQe/Main
Employee Name: polireddy
Employee Salary: 500000.0

Manager Name: saikumar
Manager Salary: 80000.0
Manager Department: microsoft company

=== Code Execution Successful ===

```

2. Create a superclass Person with properties name and age, and a method displayInfo(). Create a subclass Student that adds a property studentId and overrides the displayInfo() method. Use the super keyword to call the superclass method.

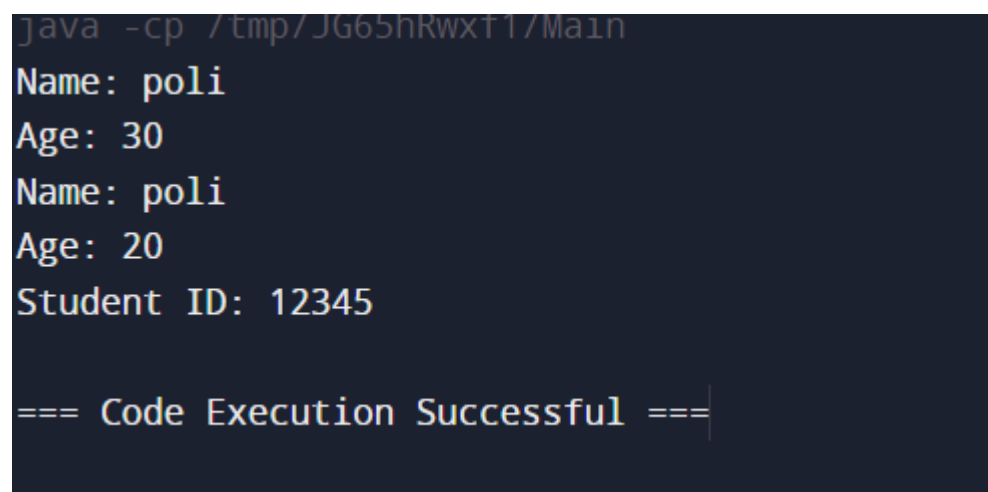
Program:

```
class Person {  
    String name;  
    int age;  
  
    Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    void displayInfo() {  
        System.out.println("Name: " + name);  
        System.out.println("Age: " + age);  
    }  
}  
  
class Student extends Person {  
    String studentId;  
  
    Student(String name, int age, String studentId) {  
        super(name, age);  
        this.studentId = studentId;  
    }  
  
    void displayInfo() {  
        super.displayInfo();  
        System.out.println("Student ID: " + studentId);  
    }  
}
```

```
}
```

```
public class Main {  
    public static void main(String[] args) {  
  
        Person person = new Person("poli", 30);  
        person.displayInfo();  
  
        Student student = new Student("", 20, "12345");  
        student.displayInfo();  
    }  
}
```

Output:



```
java -cp /tmp/JG65hRwxT1/Main  
Name: poli  
Age: 30  
Name: poli  
Age: 20  
Student ID: 12345  
  
=== Code Execution Successful ===
```

3. Create a class Vehicle with a method move(). Create subclasses Car and Bicycle, each overriding the move() method to provide specific implementations. Demonstrate the use of overridden methods.

Program:

```
class Vehicle {  
    void move() {  
        System.out.println("The vehicle is moving.");  
    }  
}
```

```
}  
}
```

```
class Car extends Vehicle {
```

```
    @Override  
    void move() {  
        System.out.println("The car is driving.");  
    }  
}
```

```
class Bicycle extends Vehicle {
```

```
    @Override  
    void move() {  
        System.out.println("The bicycle is pedaling.");  
    }  
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Vehicle vehicle = new Vehicle();  
        vehicle.move();
```

```
        Car car = new Car();  
        car.move();
```

```
        Bicycle bicycle = new Bicycle();
```

```
        bicycle.move();  
    }  
}
```

Output:

```
java -cp /tmp/BaQSTAV3C0/Main  
The vehicle is moving.  
The car is driving.  
The bicycle is pedaling.  
  
=== Code Execution Successful ===
```