**Lab Sheet 2: Create a java program using constructor with keyword ‘super’.**

**Aim:** The aim of this Java program is to demonstrate the use of constructors with the **super** keyword in a multi-level inheritance scenario. It showcases how constructors in the subclass can call the constructor of their superclass using **super**, and how data is passed through the hierarchy to create objects with initialized properties.

**Algorithm:**

1. Create a **Person** class with private fields for name and age. Define a constructor that takes name and age as parameters to initialize these fields.
2. In the **Person** class, create a method **displayInfo()** to display the name and age.
3. Create an **Employee** class that extends **Person**. This class adds an employee-specific field, **employeeId**. Define a constructor for **Employee** that takes name, age, and employeeId as parameters.
4. In the **Employee** class, create a method **displayEmployeeInfo()** that calls the **displayInfo()** method from the superclass and displays the employeeId.
5. Create a **Student** class that also extends **Person**. This class adds a student-specific field, **studentId**. Define a constructor for **Student** that takes name, age, and studentId as parameters.
6. In the **Student** class, create a method **displayStudentInfo()** that calls the **displayInfo()** method from the superclass and displays the studentId.
7. In the **MultiLevelInheritanceExample** class, create instances of **Employee** and **Student** using their respective constructors.
8. Call the **displayEmployeeInfo()** and **displayStudentInfo()** methods to display the information of the employee and student, respectively.

**Program Explanation:**

In this program, we have a class hierarchy involving **Person**, **Employee**, and **Student**, where **Employee** and **Student** inherit from **Person**.

1. The **Person** class has private fields **name** and **age** and a constructor that initializes these fields with the provided values. It also has a method **displayInfo()** to display the name and age.
2. The **Employee** class is a subclass of **Person**. It adds an **employeeId** field and has a constructor that takes name, age, and employeeId as parameters. In the **displayEmployeeInfo()** method, it calls the **displayInfo()** method from the superclass using **super** and then displays the employeeId.
3. The **Student** class is also a subclass of **Person** and is similar to the **Employee** class but with a **studentId** field and a **displayStudentInfo()** method.

In the **MultiLevelInheritanceExample** class, we create instances of **Employee** and **Student**, passing relevant information to their constructors. We then call their respective display methods to print their information.

The use of the **super** keyword in the constructors ensures that the superclass constructor is invoked before initializing the subclass-specific fields, allowing for proper object initialization and inheritance of properties.

**Top of Form**

**Program:**

class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public void displayInfo() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

}

class Employee extends Person {

private String employeeId;

public Employee(String name, int age, String employeeId) {

super(name, age);

this.employeeId = employeeId;

}

public void displayEmployeeInfo() {

super.displayInfo();

System.out.println("Employee ID: " + employeeId);

}

}

class Student extends Person {

private String studentId;

public Student(String name, int age, String studentId) {

super(name, age);

this.studentId = studentId;

}

public void displayStudentInfo() {

super.displayInfo();

System.out.println("Student ID: " + studentId);

}

}

public class MultiLevelInheritanceExample {

public static void main(String[] args) {

Employee employee = new Employee("John", 30, "EMP12345");

Student student = new Student("Alice", 20, "STU67890");

System.out.println("Employee Information:");

employee.displayEmployeeInfo();

System.out.println("\nStudent Information:");

student.displayStudentInfo();

}

}

**OUTPUT:**

Employee Information:

Name: John

Age: 30

Employee ID: EMP12345

Student Information:

Name: Alice

Age: 20

Student ID: STU67890