INHERITANCE

i) Students Details:

CODE:

```
class College {
   String collegeName = "AMRITA";
   String address = "CHENNAI, India";

   void showCollegeDetails() {
      System.out.println("College Name: " + collegeName);
      System.out.println("Address: " + address);
   }
}
class Student extends College {
   String studentName;
   int rollNumber;

   Student(String studentName, int rollNumber) {
      this.studentName = studentName;
      this.rollNumber = rollNumber;
   }

   void showStudentDetails() {
        System.out.println("Student Name: " + studentName);
        System.out.println("Roll Number: " + rollNumber);
    }
}
public class SingleInheritanceExample1 {
    public static void main(String[] args) {
        Student s1 = new Student("Rahul", 101);
        s1.showCollegeDetails();
        s1.showStudentDetails();
    }
}
```

ii)Bank Details

CODE:

```
class BankAccount {
    String accountHolder;
    double balance;

    BankAccount(String accountHolder, double balance) {
        this.accountHolder = accountHolder;
        this.balance = balance;
    }
    void showBalance() {
        System.out.println("Account Holder: " + accountHolder);
        System.out.println("Balance: $" + balance);
    }
}
class SavingsAccount extends BankAccount {
    double interestRate = 5.0;

    SavingsAccount(String accountHolder, double balance) {
        super(accountHolder, balance);
    }

    void calculateInterest() {
        double interest = (balance * interestRate) / 100;
        System.out.println("Annual Interest: $" + interest);
    }
}

public class SingleInheritanceExample2 {
    public static void main(String[] args) {
        SavingsAccount acc1 = new SavingsAccount("John Doe",
5000);
```

```
        acc1.showBalance();
        acc1.calculateInterest();
    }
}
```

MULTILEVEL INHERITANCE

i)General Details

CODE:

```
class LivingBeing {
    void breathe() {
        System.out.println("Living beings breathe.");
    }
}
class Human extends LivingBeing {
    void speak() {
        System.out.println("Humans can speak.");
    }
}
class Student extends Human {
    String name;
    int studentID;
    Student(String name, int studentID) {
        this.name = name;
        this.studentID = studentID;
    }

    void study() {
        System.out.println(name + " is studying.");
    }

    void showDetails() {
        System.out.println("Student Name: " + name);
        System.out.println("Student ID: " + studentID);
    }
}
public class MultilevelExample1 {
    public static void main(String[] args) {
        Student s1 = new Student("Rahul", 101);
        s1.breathe();
        s1.speak();
        s1.study();
        s1.showDetails();
    }
}
```

ii)Employee

CODE:

```
class Person {
    String name;
    Person(String name) {
        this.name = name;
    }
    void showPerson() {
        System.out.println("Person Name: " + name);
    }
}
class Employee extends Person {
    int employeeID;
    double salary;
    Employee(String name, int employeeID, double salary) {
        super(name);
        this.employeeID = employeeID;
        this.salary = salary;
    }
    void showEmployee() {
        System.out.println("Employee ID: " + employeeID);
        System.out.println("Salary: $" + salary);
    }
}
class Manager extends Employee {
    String department;
    Manager(String name, int employeeID, double salary, String
department) {
        super(name, employeeID, salary);
        this.department = department;
    }
    void showManager() {
        System.out.println("Department: " + department);
        System.out.println("Role: Manager");
    }
}
public class MultilevelExample2 {
    public static void main(String[] args) {
        Manager m1 = new Manager("Alice", 2001, 75000, 
"HR");

        m1.showPerson();

        m1.showEmployee();

        m1.showManager();

    }

}
```

HIERARCHICAL INHERITANCE PROGRAMS

i)Code:

```
        class Vehicle {
            private String brand;
            private String model;
             public Vehicle(String brand, String model) {
                this.brand = brand;
                this.model = model;
            }
            public void start() {
                System.out.println("Vehicle is starting.");
            }
            public void stop() {
                System.out.println("Vehicle is stopping.");
            }
            public String getBrand() {
                return brand;
            }
          public String getModel() {
```

```
        return model;
    }
}
class Car extends Vehicle {
    private int numberOfDoors ;
   public Car(String brand, String model, int
numberOfDoors) {
        super(brand, model);
        this.numberOfDoors = numberOfDoors;
    }
  public void drive() {
        System.out.println("Car is driving.");
    }
    public int getNumberOfDoors() {
        return numberOfDoors;
    }
}
class ElectricCar extends Car {
    private int batteryCapacity;
   public ElectricCar(String brand, String model, int
numberOfDoors, int batteryCapacity) {
        super(brand, model, numberOfDoors);
        this.batteryCapacity = batteryCapacity;
    }
  public void charge() {
        System.out.println("Electric car is charging.");
    }
 public int getBatteryCapacity() {
        return batteryCapacity;
    }
}
```

```
class Truck extends Vehicle {
    private double cargoCapacity;
    public Truck(String brand, String model, double
cargoCapacity) {
        super(brand, model);
        this.cargoCapacity = cargoCapacity;
    }
 public void loadCargo() {
        System.out.println("Truck is loading cargo.");
    }
  public double getCargoCapacity() {
        return cargoCapacity;
    }
}
public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Corolla", 4);
        car.start();
        car.drive();
        car.stop();
        System.out.println("Car doors: " +
car.getNumberOfDoors());
        ElectricCar electricCar = new ElectricCar("Tesla",
"Model S", 4, 100);
        electricCar.start();
        electricCar.drive();
        electricCar.charge();
        System.out.println("Battery capacity: " +
electricCar.getBatteryCapacity());
        Truck truck = new Truck("Ford", "F-150", 2000.5);
        truck.start();
```

```
            truck.loadCargo();
            truck.stop();
            System.out.println("Cargo capacity: " +
      truck.getCargoCapacity());
         }
      }
```

ii)

Code:

```
class Person {
   private String name;
   private int age;

   public Person(String name, int age) {
      this.name = name;
      this.age = age;
   }

   public void displayDetails() {
      System.out.println("Name: " + name + ", Age: " + age);
   }
}
class Student extends Person {
    private int studentId;

    private String major;

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

        super(name, age);

        this.studentId = studentId;

        this.major = major;

    }

    public void study() {

        System.out.println("Student is studying " + major);

    }

  public void displayDetails() {

        super.displayDetails();

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

    }

}

class Professor extends Person {

    private String department;

    private String researchArea;

  public Professor(String name, int age, String department,
String researchArea) {

        super(name, age);
        this.department = department;
        this.researchArea = researchArea;
    }
 public void teach() {
        System.out.println("Professor is teaching in " +
department);
    }
   public void displayDetails() {
        super.displayDetails();
        System.out.println("Department: " + department + ",
Research Area: " + researchArea);
    }
}
class TeachingAssistant extends Student {
    private String course;


    public TeachingAssistant(String name, int age, int
studentId, String major, String course) {
        super(name, age, studentId, major);
        this.course = course;
    }
    public void assist() {
        System.out.println("Teaching assistant is assisting in " +
course);
    }
public void displayDetails() {
        super.displayDetails();
        System.out.println("Course: " + course);
    }
}
public class Main2 {
    public static void main(String[] args) {
        Student student = new Student("Alice", 20, 101,
"Computer Science");
        student.displayDetails();
        student.study();
         Professor professor = new Professor("Dr. Smith", 45,
"Computer Science", "AI");
        professor.displayDetails();
        professor.teach();
        TeachingAssistant ta = new TeachingAssistant("Bob", 25,
102, "Mathematics", "Calculus");
        ta.displayDetails();
        ta.study();
        ta.assist();
    }
}
```

HYBRID INHERITANCE PROGRAMS

i)

CODE:

```
class Person {
    String name;

    Person(String name) {
        this.name = name;
    }

    void showDetails() {
        System.out.println("Name: " + name);
    }
}
class Student extends Person {
    int studentID;

    Student(String name, int studentID) {
        super(name);
        this.studentID = studentID;
```

```java
    }

    void study() {
        System.out.println(name + " is studying.");
    }
}
class Teacher extends Person {
    String subject;

    Teacher(String name, String subject) {
        super(name);
        this.subject = subject;
    }

    void teach() {
        System.out.println(name + " is teaching " + subject + ".");
    }
}
interface Assistant {
    void assist();
}
```

```
class TeachingAssistant extends Student implements Assistant
{
    TeachingAssistant(String name, int studentID) {
        super(name, studentID);
    }

    public void assist() {
        System.out.println(name + " is assisting in a lab
session.");
    }
}
public class HybridInheritanceExample1 {
    public static void main(String[] args) {
        TeachingAssistant ta = new TeachingAssistant("Alex",
101);
        ta.showDetails();
        ta.study();
        ta.assist();
    }
}
```

ii)

CODE:

```
class Vehicle {
    void startEngine() {
        System.out.println("Vehicle engine started.");
    }
}
class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving.");
    }
}
class Boat extends Vehicle {
    void sail() {
        System.out.println("Boat is sailing.");
    }
}
interface Amphibious {
    void switchMode();
}
class AmphibiousCar extends Car implements Amphibious {
    public void switchMode() {
        System.out.println("Switching between land and water
mode.");
    }
    void sail() {
        System.out.println("Amphibious car is sailing on water.");
    }
}
public class HybridInheritanceExample2 {
    public static void main(String[] args) {
        AmphibiousCar ac = new AmphibiousCar();
        ac.startEngine();
        ac.drive();
        ac.switchMode();
        ac.sail();
    }
}
```