# Week 9

1. Create a class Vehicle, write a method cost() in this class. Create two classes Bus and Train which have their own display() methods and inherit from Vehicle class. Create objects of Bus and Train class and call cost() and display() methods.

```
class Vehicle {
  public void cost() {
    System.out.println("Calculating vehicle cost...");
  }
}

class Bus extends Vehicle {
  public void display() {
    System.out.println("This is a Bus");
  }
}

class Train extends Vehicle {
  public void display() {
    System.out.println("This is a Train");
  }
}

// Main class to test
class VehicleTest {
  public static void main(String[] args) {
    Bus bus = new Bus();
    Train train = new Train();

    System.out.println("=== BUS ===");
    bus.cost();
    bus.display();

    System.out.println("\n=== TRAIN ===");
    train.cost();
    train.display();
```

```
  }
}
```

2. Create class University which has data members- name and ranking. Create class Faculty that extends University class has data member- name and method- Details(). Create a new class Department which is derived from Faculty and has data member- name, chairman and method- Details() and Display() where Display() method calls Details() methods of both Faculty and Department class in its body. Create an object of Department class to Display() method and University ranking.

```
public class University {

  String name;
  int ranking;

  public University() {
    name = "Default University";
    ranking = 0;
  }

  public University(String name, int ranking) {
    this.name = name;
    this.ranking = ranking;
  }
}

class Faculty extends University {
  String facultyName;

  public Faculty() {
    super();
    facultyName = "Default Faculty";
  }

  public Faculty(String uniName, int ranking, String facultyName) {
    super(uniName, ranking);
    this.facultyName = facultyName;
  }
```

```
    public void Details() {
        System.out.println("Faculty Name: " + facultyName);
    }
}

class Department extends Faculty {
    String departmentName;
    String chairman;

    public Department() {
        super();
        departmentName = "Default Department";
        chairman = "Unknown";
    }

    public Department(String uniName, int ranking, String facultyName, String
deptName, String chairman) {
        super(uniName, ranking, facultyName);
        this.departmentName = deptName;
        this.chairman = chairman;
    }

    public void Details() {
        System.out.println("Department Name: " + departmentName);
        System.out.println("Chairman: " + chairman);
    }

    public void Display() {
        System.out.println("University Name: " + name);
        System.out.println("University Ranking: " + ranking);
        super.Details(); // Call Faculty's Details
        Details(); // Call Department's Details
    }
}

// Main class to test
class UniversityTest {
    public static void main(String[] args) {
        Department dept = new Department("ABC University", 5, "Engineering",
"Computer Science", "Dr. Smith");
        dept.Display();
```

```
  }
}
```

3. Create class Account (Data members- Id, Account_holder_name, Address; Methodsdeposit(), withdraw()). Create two static methods in Account calculateSimpleInterest() and calculateCompoundInterest() and implement them.

```
class Account {
  int id;
  String accountHolderName;
  String address;
  double balance;

  // Constructor
  public Account(int id, String name, String address) {
    this.id = id;
    this.accountHolderName = name;
    this.address = address;
    this.balance = 0.0;
  }

  // Instance methods
  public void deposit(double amount) {
    if (amount > 0) {
      balance += amount;
      System.out.println("Deposited: ₹" + amount);
    }
  }

  public void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
      balance -= amount;
      System.out.println("Withdrawn: ₹" + amount);
    } else {
      System.out.println("Insufficient balance!");
    }
  }

  // Static methods
  public static double calculateSimpleInterest(double principal, double rate, double time) {
```

```
    return (principal * rate * time) / 100;
  }

  public static double calculateCompoundInterest(double principal, double rate,
double time) {
    return principal * Math.pow(1 + rate/100, time) - principal;
  }

  public void display() {
    System.out.println("Account ID: " + id);
    System.out.println("Account Holder: " + accountHolderName);
    System.out.println("Address: " + address);
    System.out.println("Balance: ₹" + balance);
  }
}

// Main class to test
class AccountTest {
  public static void main(String[] args) {
    Account acc = new Account(101, "John Doe", "123 Main St");

    acc.deposit(5000);
    acc.withdraw(2000);
    acc.display();

    // Using static methods without creating object
    double si = Account.calculateSimpleInterest(10000, 5, 2);
    double ci = Account.calculateCompoundInterest(10000, 5, 2);

    System.out.println("\nSimple Interest: ₹" + si);
    System.out.println("Compound Interest: ₹" + ci);
  }
}
```

4. Create two children of Account- Saving (Data Members- Min_balance; Methodsdisplay(), deposit(), withdraw()) and Current (Data Members- Max_withdrawl_limit; Methods-display(),deposit(), withdraw()) . Create constructors for both classes. Implementation of deposit() and withdraw() should be specific to Saving and Current class. Create objects of Saving and Current class and display them.

```
class Student {
    int rollNo;
    String name;
    double marks;

    // Static variable - shared by all objects
    static String schoolName = "Default School";

    // Constructor
    public Student(int rollNo, String name, double marks) {
        this.rollNo = rollNo;
        this.name = name;
        this.marks = marks;
    }

    // Static method to change school name
    public static void changeSchoolName(String newName) {
        schoolName = newName;
        System.out.println("School name changed to: " + newName);
    }

    // Instance method
    public void display() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Marks: " + marks);
        System.out.println("School: " + schoolName);
        System.out.println("--------------------");
    }
}

// Main class to test
class StudentTest {
    public static void main(String[] args) {
        // Create students
        Student student1 = new Student(1, "Alice", 85.5);
        Student student2 = new Student(2, "Bob", 92.0);
        Student student3 = new Student(3, "Charlie", 78.5);

        System.out.println("=== INITIAL SCHOOL NAME ===");
        student1.display();
```

```
        student2.display();

        // Change school name using static method
        Student.changeSchoolName("olmpian public school");

        System.out.println("=== AFTER SCHOOL NAME CHANGE ===");
        student1.display();
        student2.display();
        student3.display();

        // All objects share the same static variable
        System.out.println("All students share the same school: " +
Student.schoolName);
    }
}
```

5. Create a class Shape with a method area(). Create two derived classes Rectangle and Circle that extend Shape. Each class should override the area() method to calculate the area of the respective shape. Create objects of Rectangle and Circle and call their area() methods.

```
class Shape {
    public double area() {
        return 0.0;
    }
}

class Rectangle extends Shape {
    double length;
    double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double area() {
        return length * width;
    }
}
```

```
class Circle extends Shape {
  double radius;

  public Circle(double radius) {
    this.radius = radius;
  }

  @Override
  public double area() {
    return Math.PI * radius * radius;
  }
}

// Main class to test
class ShapeTest {
  public static void main(String[] args) {
    Rectangle rect = new Rectangle(5, 10);
    Circle circle = new Circle(7);

    System.out.println("Rectangle Area: " + rect.area());
    System.out.println("Circle Area: " + circle.area());
  }
}
```

6. Create a class Employee with data members: name, salary, and a method showDetails(). Create a class Manager that extends Employee with an additional data member department. Override the showDetails() method in Manager to display all details, including department. Create an object of Manager and call showDetails().

```
class Employee {
  String name;
  double salary;

  public Employee(String name, double salary) {
    this.name = name;
    this.salary = salary;
  }

  public void showDetails() {
    System.out.println("Name: " + name);
    System.out.println("Salary: ₹" + salary);
```

```
    }
  }

  class Manager extends Employee {
    String department;

    public Manager(String name, double salary, String department) {
      super(name, salary);
      this.department = department;
    }

    @Override
    public void showDetails() {
      super.showDetails(); // Call parent method
      System.out.println("Department: " + department);
      System.out.println("Position: Manager");
    }
  }

  // Main class to test
  class EmployeeTest {
    public static void main(String[] args) {
      Manager mgr = new Manager("John Manager", 75000, "IT");
      mgr.showDetails();
    }
  }
```

7. Create an abstract class Appliance with data members brand, power and abstract methods turnOn() and turnOff(). Create two derived classes WashingMachine and Refrigerator that provide their own implementations of turnOn() and turnOff(). Create objects of WashingMachine and Refrigerator and call their methods.

```
abstract class Appliance {
  String brand;
  double power;

  public Appliance(String brand, double power) {
    this.brand = brand;
    this.power = power;
  }
```

```
    // Abstract methods
    public abstract void turnOn();
    public abstract void turnOff();

    // Concrete method
    public void displayInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Power: " + power + " watts");
    }
}

class WashingMachine extends Appliance {
    public WashingMachine(String brand, double power) {
        super(brand, power);
    }

    @Override
    public void turnOn() {
        System.out.println("Washing Machine is starting...");
        System.out.println("Water filling... Drum rotating...");
    }

    @Override
    public void turnOff() {
        System.out.println("Washing Machine is stopping...");
        System.out.println("Drain water... Spin cycle complete.");
    }
}

class Refrigerator extends Appliance {
    public Refrigerator(String brand, double power) {
        super(brand, power);
    }

    @Override
    public void turnOn() {
        System.out.println("Refrigerator is starting...");
        System.out.println("Compressor running... Cooling initiated.");
    }

    @Override
```

```
  public void turnOff() {
    System.out.println("Refrigerator is stopping...");
    System.out.println("Compressor stopped... Defrost cycle complete.");
  }
}

// Main class to test
class ApplianceTest {
  public static void main(String[] args) {
    WashingMachine wm = new WashingMachine("Samsung", 500);
    Refrigerator fridge = new Refrigerator("LG", 200);

    System.out.println("=== WASHING MACHINE ===");
    wm.displayInfo();
    wm.turnOn();
    wm.turnOff();

    System.out.println("\n=== REFRIGERATOR ===");
    fridge.displayInfo();
    fridge.turnOn();
    fridge.turnOff();
  }
  }
```

8. Create a class MathOperations with two static methods: findGCD(int a, int b) to calculate the greatest common divisor and findLCM(int a, int b) to calculate the least common multiple. Call these methods without creating an object of MathOperations.

```
class MathOperations {

  // Static method to find GCD
  public static int findGCD(int a, int b) {
    while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
    }
    return a;
  }

  // Static method to find LCM
```

```
    public static int findLCM(int a, int b) {
        return (a * b) / findGCD(a, b);
    }
}

// Main class to test
class MathTest {
    public static void main(String[] args) {
        // Calling static methods without creating object
        int gcd = MathOperations.findGCD(48, 18);
        int lcm = MathOperations.findLCM(12, 18);

        System.out.println("GCD of 48 and 18: " + gcd);
        System.out.println("LCM of 12 and 18: " + lcm);

        // Test with more numbers
        System.out.println("GCD of 54 and 24: " + MathOperations.findGCD(54, 24));
        System.out.println("LCM of 8 and 10: " + MathOperations.findLCM(8, 10));
    }
}
```

9. Create a class Student with data members rollNo, name, marks. Add a static variable schoolName. Create static method changeSchoolName() to update schoolName. Demonstrate how the static variable is shared among all objects.

```
class Student {
    int rollNo;
    String name;
    double marks;

    // Static variable - shared by all objects
    static String schoolName = "Default School";

    // Constructor
    public Student(int rollNo, String name, double marks) {
        this.rollNo = rollNo;
        this.name = name;
        this.marks = marks;
    }

    // Static method to change school name
```

```
    public static void changeSchoolName(String newName) {
        schoolName = newName;
        System.out.println("School name changed to: " + newName);
    }

    // Instance method
    public void display() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Marks: " + marks);
        System.out.println("School: " + schoolName);
        System.out.println("--------------------");
    }
}

// Main class to test
class StudentTest {
    public static void main(String[] args) {
        // Create students
        Student student1 = new Student(1, "Alice", 85.5);
        Student student2 = new Student(2, "Bob", 92.0);
        Student student3 = new Student(3, "Charlie", 78.5);

        System.out.println("=== INITIAL SCHOOL NAME ===");
        student1.display();
        student2.display();

        // Change school name using static method
        Student.changeSchoolName("olmpian public school");

        System.out.println("=== AFTER SCHOOL NAME CHANGE ===");
        student1.display();
        student2.display();
        student3.display();

        // All objects share the same static variable
        System.out.println("All students share the same school: " +
Student.schoolName);
    }
}
```