Assignment No. 2

OOPS 4 Pillar Based

1. Inheritance (5 Questions)

1. Create a base class Animal with a method makeSound(). Create a subclass Dog that overrides makeSound() to print "Bark".

```java
class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Bark");
    }
}

public class Inheritance1 {
    public static void main(String[] args) {
        Animal a = new Animal();
        a.makeSound();

        Dog d = new Dog();
        d.makeSound();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac Inheritance1.java
PS D:\CDAC\Java\AssignmentNo2> java Inheritance1
Animal makes a sound
Bark
```

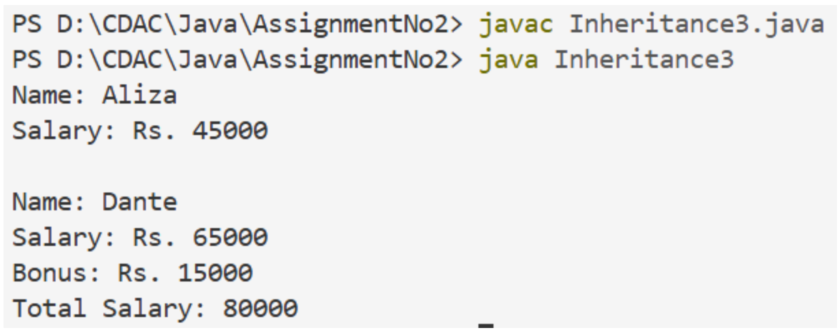
2. Create a base class Vehicle with properties brand and speed. Create a subclass Car that adds fuelType and a method displayCarDetails().

```java
class Vehicle {
    String brand;
    double speed;

    Vehicle(String brand, double speed) {
        this.brand = brand;
        this.speed = speed;
    }
    public void displayCarDetails() {
        System.out.println("Brand: " + brand + "\nSpeed (km/h): " +
speed);
    }
}

class Car extends Vehicle {
    String fuelType;

    Car(String brand, double speed, String fuelType) {
        super(brand, speed);
        this.fuelType = fuelType;
    }

    public void displayCarDetails() {
        System.out.println("Brand: " + brand + "\nSpeed (km/h): " +
speed + "\nFuel type: " + fuelType);
    }
}

public class Inheritance2 {
    public static void main(String[] args) {
        Vehicle v = new Vehicle("Toyota", 100);
        v.displayCarDetails();

        Car c = new Car("Hyundai", 150, "Petrol");
        c.displayCarDetails();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac Inheritance2.java
PS D:\CDAC\Java\AssignmentNo2> java Inheritance2
Brand: Toyota
Speed (km/h): 100.0
Brand: Hyundai
Speed (km/h): 150.0
Fuel type: Petrol
```

3. Create a base class Employee with attributes name and salary. Create a subclass Manager that adds bonus. Write a method to calculate the total salary.

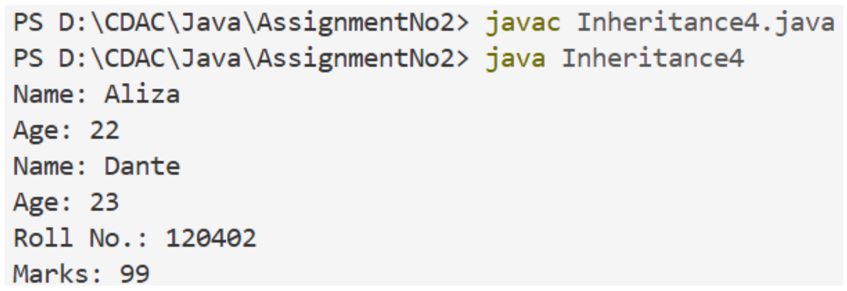
```
class Employee {
    String name;
    int salary;

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

    public void displayInfo() {
        System.out.println("Name: " + name + "\nSalary: Rs. " +
salary + "\n");
    }
}

class Manager extends Employee {
    int bonus;

    Manager(String name, int salary, int bonus) {
        super(name, salary);
        this.bonus = bonus;
    }

    public int totalSalary() {
        return salary + bonus;
    }

    @Override
    public void displayInfo() {
        System.out.println("Name: " + name + "\nSalary: Rs. " +
salary + "\nBonus: Rs. " + bonus + "\nTotal Salary: "
                + totalSalary());
    }
}

public class Inheritance3 {
    public static void main(String[] args) {
        Employee e = new Employee("Aliza", 45000);
        e.displayInfo();

        Manager m = new Manager("Dante", 65000, 15000);
        m.displayInfo();
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Inheritance3.java
PS D:\CDAC\Java\AssignmentNo2> java Inheritance3
Name: Aliza
Salary: Rs. 45000

Name: Dante
Salary: Rs. 65000
Bonus: Rs. 15000
Total Salary: 80000
```

4. Write a Java program where a Person class has name and age. Create a subclass Student that adds rollNumber and marks.

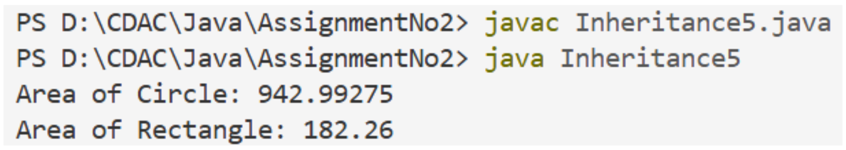
```java
class Person {
    String name;
    int age;

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

    public void displayInfo() {
        System.out.println("Name: " + name + "\nAge: " + age);
    }
}

class Student extends Person {
    int rollNo;
    int marks;

    Student(String name, int age, int rollNo, int marks) {
        super(name, age);
        this.rollNo = rollNo;
        this.marks = marks;
    }

    public void displayInfo() {
        System.out.println("Name: " + name + "\nAge: " + age +
"\nRoll No.: " + rollNo + "\nMarks: " + marks);
    }
}

public class Inheritance4 {
    public static void main(String[] args) {
        Person p = new Person("Aliza", 22);
        p.displayInfo();
        Student s = new Student("Dante", 23, 120402, 99);
        s.displayInfo();
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Inheritance4.java
PS D:\CDAC\Java\AssignmentNo2> java Inheritance4
Name: Aliza
Age: 22
Name: Dante
Age: 23
Roll No.: 120402
Marks: 99
```

5. Create a base class Shape with a method area(). Create subclasses Circle and Rectangle that override area() to calculate their respective areas.

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

class Circle extends Shape {
    public double calculateArea(double radius) {
        return (2 * 3.142 * radius * radius);
    }
}

class Rectangle extends Shape {
    public double calculateArea(double length, double breadth) {
        return (2 * length + 2 * breadth);
    }
}

public class Inheritance5 {
    public static void main(String[] args) {
        Circle c = new Circle();
        System.out.println("Area of Circle: " +
c.calculateArea(12.25));

        Rectangle r = new Rectangle();
        System.out.println("Area of Rectangle: " +
r.calculateArea(62.39, 28.74));
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Inheritance5.java
PS D:\CDAC\Java\AssignmentNo2> java Inheritance5
Area of Circle: 942.99275
Area of Rectangle: 182.26
```

2. Encapsulation (5 Questions)

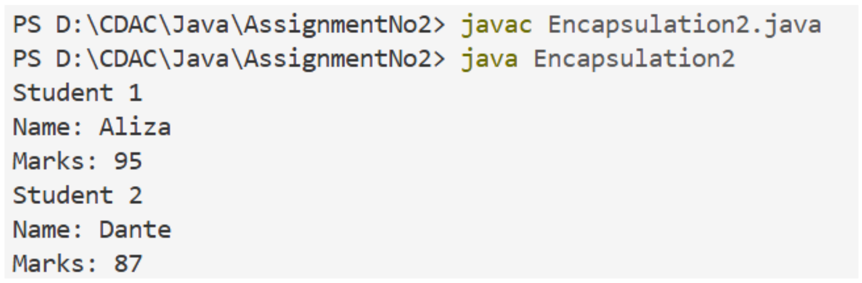
6. Create a class BankAccount with private attributes accountNumber and balance. Use getters and setters to access and modify them.

```java
class BankAccount {
    private long accountNumber;
    private long balance;

    public long getAccountNo() {
        return accountNumber;
    }

    public void setAccountNo(long accountNumber) {
        this.accountNumber = accountNumber;
    }

    public long getBalance() {
        return balance;
    }

    public void setBalance(long balance) {
        this.balance = balance;
    }
}

public class Encapsulation1 {
    public static void main(String[] args) {
        BankAccount acc1 = new BankAccount();
        acc1.setAccountNo(2501648933321l);
        acc1.setBalance(25000000l);
        System.out.println("Account 1\nAccount Number: " +
acc1.getAccountNo() + "\nBalance: Rs.  " + acc1.getBalance());

        BankAccount acc2 = new BankAccount();
        acc2.setAccountNo(2501648956745l);
        acc2.setBalance(50000l);
        System.out.println("Account 2\nAccount Number: " +
acc2.getAccountNo() + "\nBalance: Rs.  " + acc2.getBalance());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac Encapsulation1.java
PS D:\CDAC\Java\AssignmentNo2> java Encapsulation1
Account 1
Account Number: 2501648933321
Balance: Rs. 25000000
Account 2
Account Number: 2501648956745
Balance: Rs. 50000
```

7. Write a Java program to create a Student class with private variables name and marks. Use getters to retrieve and setters to modify the values.

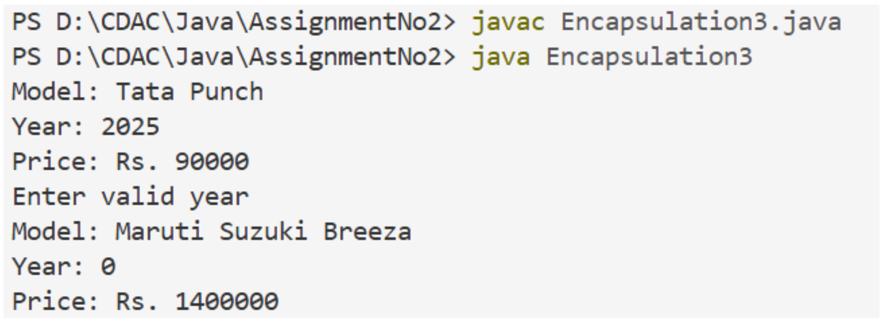
```java
class Student {
    private String name;
    private int marks;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getMarks() {
        return marks;
    }

    public void setMarks(int marks) {
        this.marks = marks;
    }
}

public class Encapsulation2 {
    public static void main(String[] args) {
        Student stud1 = new Student();
        stud1.setName("Aliza");
        stud1.setMarks(95);
        System.out.println("Student 1\nName: " + stud1.getName() +
"\nMarks: " + stud1.getMarks());

        Student stud2 = new Student();
        stud2.setName("Dante");
        stud2.setMarks(87);
        System.out.println("Student 2\nName: " + stud2.getName() +
"\nMarks: " + stud2.getMarks());
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Encapsulation2.java
PS D:\CDAC\Java\AssignmentNo2> java Encapsulation2
Student 1
Name: Aliza
Marks: 95
Student 2
Name: Dante
Marks: 87
```

8. Create a class Car with private variables model, year, and price. Provide public methods to get and set values while ensuring year is not negative.

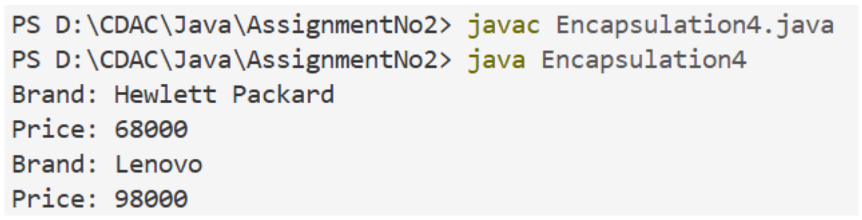
```
class Car {
    private String model;
    private int year;
    private long price;

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getYear() {
        if (year > 0) {
            return year;
        } else {
            return 0;
        }
    }

    public void setYear(int year) {
        if (year > 0) {
            this.year = year;
        } else {
            System.out.println("Enter valid year");
        }
    }

    public long getPrice() {
        return price;
    }

    public void setPrice(long price) {
        this.price = price;
    }
}

public class Encapsulation3 {
    public static void main(String[] args) {
        Car c1 = new Car();
        c1.setModel("Tata Punch");
        c1.setYear(2025);
        c1.setPrice(90000l);
        System.out.println("Model: " + c1.getModel() + "\nYear: " +
c1.getYear() + "\nPrice: Rs. " + c1.getPrice());
        Car c2 = new Car();
        c2.setModel("Maruti Suzuki Breeza");
        c2.setYear(-2025);
        c2.setPrice(1400000l);
        System.out.println("Model: " + c2.getModel() + "\nYear: " +
c2.getYear() + "\nPrice: Rs. " + c2.getPrice());
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Encapsulation3.java
PS D:\CDAC\Java\AssignmentNo2> java Encapsulation3
Model: Tata Punch
Year: 2025
Price: Rs. 90000
Enter valid year
Model: Maruti Suzuki Breeza
Year: 0
Price: Rs. 1400000
```

9. Write a Java program for a Laptop class with private attributes brand and price. Ensure price cannot be set below zero using validation inside the setter method.

```
class Laptop {
    private String brand;
    private long price;

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public long getPrice() {
        if (price > 0) {
            return price;
        } else {
            return 0;
        }
    }

    public void setPrice(long price) {
        if (price > 0) {
            this.price = price;
        } else {
            System.out.println("Enter valid price");
        }
    }
}

public class Encapsulation4 {
    public static void main(String[] args) {
        Laptop l1 = new Laptop();
        l1.setBrand("Hewlett Packard");
        l1.setPrice(68000l);
        System.out.println("Brand: " + l1.getBrand() + "\nPrice: " +
l1.getPrice());

        Laptop l2 = new Laptop();
        l2.setBrand("Lenovo");
        l2.setPrice(98000l);
        System.out.println("Brand: " + l2.getBrand() + "\nPrice: " +
l2.getPrice());
    }
}
```

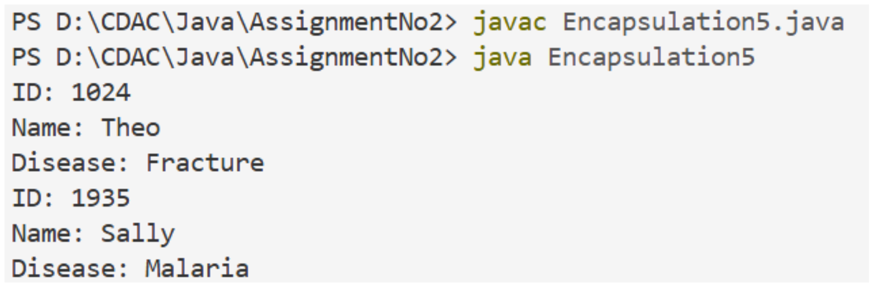


```
PS D:\CDAC\Java\AssignmentNo2> javac Encapsulation4.java
PS D:\CDAC\Java\AssignmentNo2> java Encapsulation4
Brand: Hewlett Packard
Price: 68000
Brand: Lenovo
Price: 98000
```

10. Create a Patient class with private attributes id, name, and disease. Provide methods to set and get details and restrict modification of id once assigned.

```
class Patient {
    private final int id;
    private String name;
    private String disease;

    public Patient(int id) {
        this.id = id;
```

```java
    }

    public int getID() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDisease() {
        return disease;
    }

    public void setDisease(String disease) {
        this.disease = disease;
    }
}

public class Encapsulation5 {
    public static void main(String[] args) {
        Patient p1 = new Patient(1024);
        p1.setName("Theo");
        p1.setDisease("Fracture");
        System.out.println("ID: " + p1.getID() + "\nName: " +
p1.getName() + "\nDisease: " + p1.getDisease());

        Patient p2 = new Patient(1935);
        p2.setName("Sally");
        p2.setDisease("Malaria");
        System.out.println("ID: " + p2.getID() + "\nName: " +
p2.getName() + "\nDisease: " + p2.getDisease());
    }
}
```



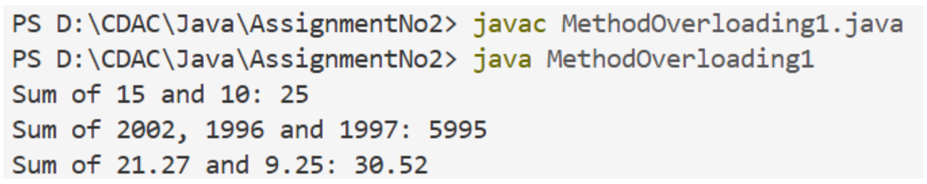
```
PS D:\CDAC\Java\AssignmentNo2> javac Encapsulation5.java
PS D:\CDAC\Java\AssignmentNo2> java Encapsulation5
ID: 1024
Name: Theo
Disease: Fracture
ID: 1935
Name: Sally
Disease: Malaria
```

3. Polymorphism (5 Questions)

(A) Compile-Time Polymorphism (Method Overloading)

11. Create a MathOperations class with overloaded add() methods: one for two integers, another for three integers, and one for two double values.

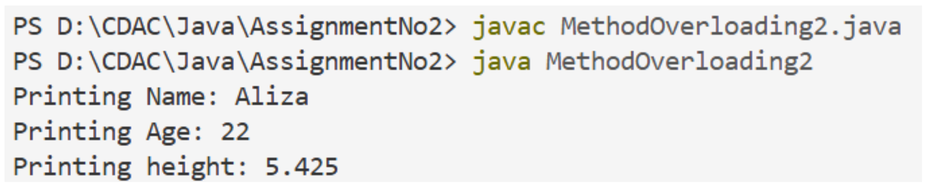
```
class MathOperations {
    public void add(int a, int b) {
        System.out.println("Sum of " + a + " and " + b + ": " + (a +
b));
    }

    public void add(int a, int b, int c) {
        System.out.println("Sum of " + a + ", " + b + " and " + c +
": " + (a + b + c));
    }

    public void add(double a, double b) {
        System.out.println("Sum of " + a + " and " + b + ": " + (a +
b));
    }
}

public class MethodOverloading1 {
    public static void main(String[] args) {
        MathOperations m1 = new MathOperations();
        m1.add(15, 10);
        MathOperations m2 = new MathOperations();
        m2.add(2002, 1996, 1997);
        MathOperations m3 = new MathOperations();
        m3.add(21.27, 9.25);
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverloading1.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverloading1
Sum of 15 and 10: 25
Sum of 2002, 1996 and 1997: 5995
Sum of 21.27 and 9.25: 30.52
```

12. Write a Java program to create a class Printer that has multiple overloaded print() methods for String, int, and double values.

```
class Printer {
    public void print(String name) {
        System.out.println("Printing Name: " + name);
    }
    public void print(int age) {
        System.out.println("Printing Age: " + age);
    }

    public void print(double height) {
        System.out.println("Printing height: " + height);
    }
}

public class MethodOverloading2 {
    public static void main(String[] args) {
        Printer p1 = new Printer();
        p1.print("Aliza");
        Printer p2 = new Printer();
        p2.print(22);
        Printer p3 = new Printer();
        p3.print(5.425);
    }
}
```

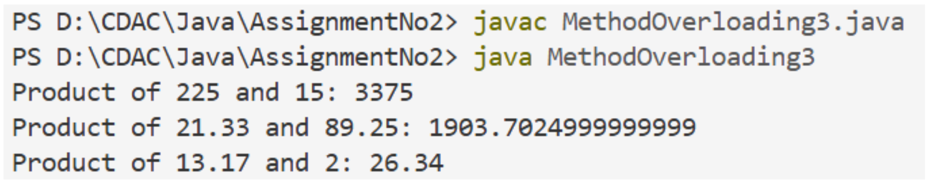


```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverloading2.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverloading2
Printing Name: Aliza
Printing Age: 22
Printing height: 5.425
```

13. Create a Calculator class with overloaded multiply() methods to accept integers, doubles, and a mix of both.

```
class Calculator {
    public void multiply(int a, int b) {
        System.out.println("Product of " + a + " and " + b + ": " +
(a * b));
    }

    public void multiply(double a, double b) {
        System.out.println("Product of " + a + " and " + b + ": " +
(a * b));
    }

    public void multiply(double a, int b) {
        System.out.println("Product of " + a + " and " + b + ": " +
(a * b));
    }
}
```

```java
public class MethodOverloading3 {
    public static void main(String[] args) {
        Calculator m1 = new Calculator();
        m1.multiply(225, 15);
        Calculator m2 = new Calculator();
        m2.multiply(21.33, 89.25);
        Calculator m3 = new Calculator();
        m3.multiply(13.17, 2);
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverloading3.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverloading3
Product of 225 and 15: 3375
Product of 21.33 and 89.25: 1903.7024999999999
Product of 13.17 and 2: 26.34
```

14. Write a Java program where a Shape class has overloaded draw() methods, accepting different numbers of parameters to draw different shapes.

```java
class Shape {
    public void draw(int a, int b, int c) {
        System.out.println("This is a triangle. Perimeter: " + (a +
b + c));
    }

    public void draw(int a, int b) {
        System.out.println("This is a rectangle. Perimeter: " + (2 *
a + 2 * b));
    }

    public void draw(int a) {
        System.out.println("This is a square. Perimeter: " + (4 *
a));
    }
}

public class MethodOverloading4 {
    public static void main(String[] args) {
        Shape m1 = new Shape();
        m1.draw(25, 15);
        Shape m2 = new Shape();
        m2.draw(89);
        Shape m3 = new Shape();
        m3.draw(13, 5, 12);
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverloading4.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverloading4
This is a rectangle. Perimeter: 80
This is a square. Perimeter: 356
This is a triangle. Perimeter: 30
```

15. Create a class CurrencyConverter that has overloaded methods to convert different currencies (INR to USD, INR to EUR, etc.).

```java
class CurrencyConverter {
    public void convert(double a) {
        System.out.println("INR to USR: " + (a * 0.011));
    }

    public void convert(double a, String b) {
        if (b == "KRW") {
            System.out.println("INR to KRW: " + (a * 15.76));
        } else if (b == "THB") {
            System.out.println("INR to THB: " + (a * 0.36));
        } else if (b == "CNY") {
            System.out.println("INR to CNY: " + (a * 0.081));
        }
    }
}

public class MethodOverloading5 {
    public static void main(String[] args) {
        CurrencyConverter m1 = new CurrencyConverter();
        m1.convert(250);
        CurrencyConverter m2 = new CurrencyConverter();
        m2.convert(12000, "THB");
        CurrencyConverter m3 = new CurrencyConverter();
        m3.convert(2500, "KRW");
        CurrencyConverter m4 = new CurrencyConverter();
        m4.convert(999, "CNY");
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverloading5.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverloading5
INR to USR: 2.75
INR to THB: 4320.0
INR to KRW: 39400.0
INR to CNY: 80.919
```

(B) Runtime Polymorphism (Method Overriding)

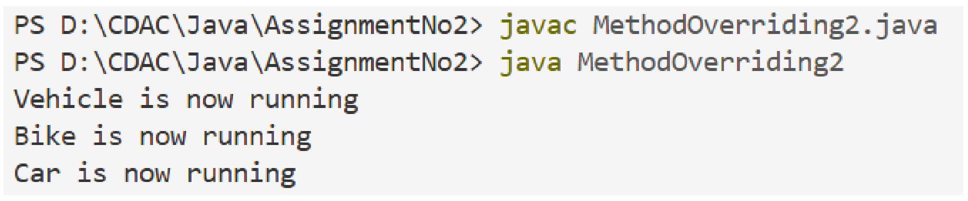
16. Create a base class Animal with speak() method. Create subclasses Dog and Cat that override speak() to print different sounds.

```java
class Animal {
    public void speak() {
        System.out.println("Animal speaks");
    }
}

class Dog extends Animal {
    @Override
    public void speak() {
        System.out.println("Dog speaks woof-woof");
    }
}

class Cat extends Animal {
    @Override
    public void speak() {
        System.out.println("Cat speaks meow-meow");
    }
}

public class MethodOverriding1 {
    public static void main(String[] args) {
        Animal a = new Animal();
        a.speak();

        Dog d = new Dog();
        d.speak();

        Cat c = new Cat();
        c.speak();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverriding1.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverriding1
Animal speaks
Dog speaks woof-woof
Cat speaks meow-meow
```

17. Write a Java program where a Vehicle class has a run() method. Create subclasses Bike and Car that override run() with specific messages.

```java
class Vehicle {
    public void run() {
        System.out.println("Vehicle is now running");
    }
}

class Bike extends Vehicle {
    @Override
    public void run() {
        System.out.println("Bike is now running");
    }
}

class Car extends Vehicle {
    @Override
    public void run() {
        System.out.println("Car is now running");
    }
}

public class MethodOverriding2 {
    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.run();

        Bike b = new Bike();
        b.run();

        Car c = new Car();
        c.run();
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverriding2.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverriding2
Vehicle is now running
Bike is now running
Car is now running
```

18. Create a Bank class with a method getInterestRate(). Create subclasses SBI, HDFC, and ICICI that override the method with their respective interest rates.

```java
class Bank {
    public void getInterestRate() {
        System.out.println("Banks have interest rate");
    }
}

class SBI extends Bank {
    @Override
    public void getInterestRate() {
        System.out.println("SBI interest rate: 2.89%");
    }
}

class HDFC extends Bank {
    @Override
    public void getInterestRate() {
        System.out.println("HDFC interest rate: 3%");
    }
}

class ICICI extends Bank {
    @Override
    public void getInterestRate() {
        System.out.println("ICICI interest rate: 3.5%");
    }
}

public class MethodOverriding3 {
    public static void main(String[] args) {
        Bank a = new Bank();
        a.getInterestRate();

        SBI s = new SBI();
        s.getInterestRate();

        HDFC h = new HDFC();
        h.getInterestRate();

        ICICI i = new ICICI();
        i.getInterestRate();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverriding3.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverriding3
Banks have interest rate
SBI interest rate: 2.89%
HDFC interest rate: 3%
ICICI interest rate: 3.5%
```

19. Write a Java program where a Phone class has a method call(). Create subclasses Smartphone and Landline that override call() differently.

```java
class Phone {
    public void call() {
        System.out.println("Phone is used for calling");
    }
}

class Smartphone extends Phone {
    @Override
    public void call() {
        System.out.println("Smartphone has touchscreen");
    }
}

class Landline extends Phone {
    @Override
    public void call() {
        System.out.println("Landline has keypad");
    }
}

public class MethodOverriding4 {
    public static void main(String[] args) {
        Phone p = new Phone();
        p.call();

        Smartphone s = new Smartphone();
        s.call();

        Landline l = new Landline();
        l.call();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverriding4.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverriding4
Phone is used for calling
Smartphone has touchscreen
Landline has keypad
```

20. Create a Browser class with a method openWebsite(). Create subclasses Chrome and Firefox that override openWebsite() with specific implementation details.

```java
class Website {
    public void openWebsite() {
        System.out.println("Browser is used for opening websites");
    }
}

class Chrome extends Website {
    @Override
    public void openWebsite() {
        System.out.println("Chrome is opening Amazon");
    }
}

class Firefox extends Website {
    @Override
    public void openWebsite() {
        System.out.println("Firefox is opening Anna's Archive");
    }
}

public class MethodOverriding5 {
    public static void main(String[] args) {
        Website w = new Website();
        w.openWebsite();

        Chrome c = new Chrome();
        c.openWebsite();

        Firefox f = new Firefox();
        f.openWebsite();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac MethodOverriding5.java
PS D:\CDAC\Java\AssignmentNo2> java MethodOverriding5
Browser is used for opening websites
Chrome is opening Amazon
Firefox is opening Anna's Archive
```

4. Abstraction (5 Questions)

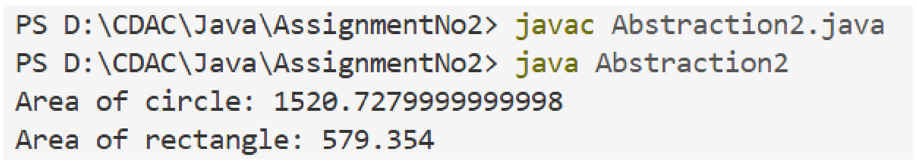
21. Create an abstract class Vehicle with an abstract method start(). Create subclasses Car and Bike that provide their own implementation of start().

```
abstract class Vehicle {
    abstract void start();
}

class Car extends Vehicle {
    public void start() {
        System.out.println("Car starting.....");
    }
}

class Bike extends Vehicle {
    public void start() {
        System.out.println("Bike starting.....");
    }
}

public class Abstraction1 {
    public static void main(String[] args) {
        Vehicle c = new Car();
        c.start();

        Vehicle b = new Bike();
        b.start();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac Abstraction1.java
PS D:\CDAC\Java\AssignmentNo2> java Abstraction1
Car starting.....
Bike starting.....
```

22. Write a Java program with an abstract class Shape that has an abstract method calculateArea(). Implement it in Circle and Rectangle classes.

```
abstract class Shape {
    abstract double calculateArea();
}

class Circle extends Shape {
    double a;

    Circle(double a) {
        this.a = a;
    }

    public double calculateArea() {
        return (3.142 * a * a);
    }
}

class Rectangle extends Shape {
    double l;
    double b;

    Rectangle(double l, double b) {
        this.l = l;
        this.b = b;
    }

    public double calculateArea() {
        return (l * b);
    }
}

public class Abstraction2 {
    public static void main(String[] args) {
        Shape c = new Circle(22);
        System.out.println("Area of circle: " + c.calculateArea());

        Shape r = new Rectangle(22.3, 25.98);
        System.out.println("Area of rectangle: " +
r.calculateArea());
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Abstraction2.java
PS D:\CDAC\Java\AssignmentNo2> java Abstraction2
Area of circle: 1520.7279999999998
Area of rectangle: 579.354
```

23. Create an abstract class Payment with an abstract method payAmount(). Create subclasses CreditCardPayment and UPIPayment that implement it differently.

```
abstract class Payment {
    abstract void payAmount();
}

class CreditCardPayment extends Payment {
    void payAmount() {
        System.out.println("Credit Card Payment Method");
    }
}

class UPIPayment extends Payment {
    void payAmount() {
        System.out.println("UPI Payment Method");
    }
}

public class Abstraction3 {
    public static void main(String[] args) {
        Payment cc = new CreditCardPayment();
        cc.payAmount();

        Payment upi = new UPIPayment();
        upi.payAmount();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac Abstraction3.java
PS D:\CDAC\Java\AssignmentNo2> java Abstraction3
Credit Card Payment Method
UPI Payment Method
```

24. Write a Java program with an abstract class Employee that has an abstract method calculateSalary(). Implement it in FullTimeEmployee and PartTimeEmployee classes.

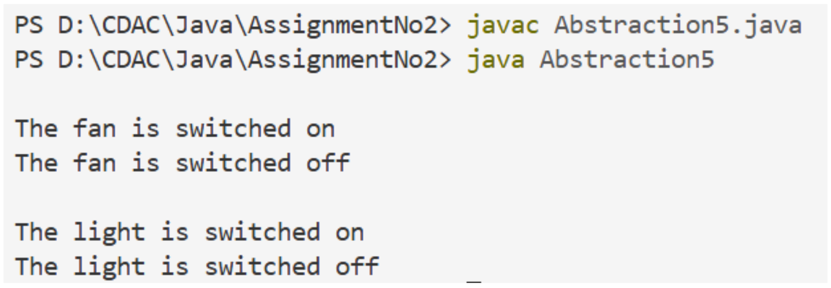
```
abstract class Employee {
    abstract void calculateSalary();
}

class FullTimeEmployee extends Employee {
    public void calculateSalary() {
        System.out.println("Full time employee salary: Rs. 45000");
    }
}

class PartTimeEmployee extends Employee {
    public void calculateSalary() {
        System.out.println("Part time employee salary: Rs. 15000");
    }
}

public class Abstraction4 {
    public static void main(String[] args) {
        Employee ft = new FullTimeEmployee();
```

```java
        ft.calculateSalary();

        Employee pt = new PartTimeEmployee();
        pt.calculateSalary();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo2> javac Abstraction4.java
PS D:\CDAC\Java\AssignmentNo2> java Abstraction4
Full time employee salary: Rs. 45000
Part time employee salary: Rs. 15000
```

25. Create an abstract class Appliance with abstract methods turnOn() and turnOff(). Implement these in Fan and Light classes.

```java
abstract class Appliance {
    abstract void turnOn();

    abstract void turnOff();
}

class Fan extends Appliance {
    public void turnOn() {
        System.out.println("\nThe fan is switched on");
    }

    public void turnOff() {
        System.out.println("The fan is switched off");
    }
}

class Light extends Appliance {
    public void turnOn() {
        System.out.println("\nThe light is switched on");
    }

    public void turnOff() {
        System.out.println("The light is switched off");
    }
}

public class Abstraction5 {
    public static void main(String[] args) {
        Appliance fan = new Fan();
        fan.turnOn();
        fan.turnOff();
        Appliance light = new Light();
        light.turnOn();
        light.turnOff();
    }
}
```



```
PS D:\CDAC\Java\AssignmentNo2> javac Abstraction5.java
PS D:\CDAC\Java\AssignmentNo2> java Abstraction5

The fan is switched on
The fan is switched off

The light is switched on
The light is switched off
```