

Week: 4**1. Write a Java program to implement the concept of****inheritance.**Code: class Animal {

String name;

public Animal(String

name) {this.name =

name;

}

public void eat() {

System.out.println(name + " is

eating.");

}

}

class Dog extends

Animal {public

Dog(String name) {

super(name);

}

public void bark() {

System.out.println(name + " is

barking.");

}

}

class Cat extends

Animal {public

Cat(String name) {

super(name);

}

public void meow() {

System.out.println(name + " is

meowing.");

}

}

public class InheritanceExample {

public static void main(String[]

args) {

Dog myDog = new

Dog("Buddy"); Cat myCat =

new Cat("Whiskers");

myDog.eat();

myCat.eat();

myDog.bark();

myCat.meow();

}

```

}
Buddy is eating.
Whiskers is eating.
Buddy is barking.
Whiskers is meowing.
PS D:\RN_1\Coding\JAVA\Assignments\src>

```

2. Write a Java program to show method overloading.

```

3. Code: public class MethodOverloadingExample {
    public int add(int a, int
        b) {return a + b;
    }
    public int add(int a, int b, int
        c) {return a + b + c;
    }
    public double add(double a, double
        b) {return a + b;
    }
    public String concatenate(String str1, String
        str2) {return str1 + str2;
    }
    public static void main(String[] args) {
        MethodOverloadingExample example = new
        MethodOverloadingExample();
        System.out.println("Sum (int): " + example.add(7, 13));
        System.out.println("Sum (int): " + example.add(5, 10, 15));
        System.out.println("Sum (double): " + example.add(3.5, 2.5));
        System.out.println("Concatenation: " + example.concatenate("Hello, ", "world!"));
    }

    Sum (int): 20
    Sum (int): 30
    Sum (double): 6.0
    Concatenation: Hello, world!
    PS C:\Users\User\Desktop\Java\practice>
}

```

4. Write a Java program to show method overriding.

```

Code: class Animal {
    public void sound() {
        System.out.println("Animal makes a
        sound");
    }
}
class Dog extends
    Animal {public void
    sound() {
        System.out.println("Dog barks");
    }
}

```

```
}  
public class  
MethodOverridingExample {public  
static void main(String[] args) {  
    Animal animal = new  
    Animal();Dog dog = new  
    Dog(); animal.sound();  
    dog.sound();  
}  
}
```

Animal makes a sound
Dog barks

5. Write a Java program to show method**hiding.Code:** class Animal {

```

    public static void eat() {
        System.out.println("Animal is
        eating");
    }
}

```

class Dog extends

```

    Animal {public static
    void eat() {
        System.out.println("Dog is eating");
    }
}

```

public class MethodHidingExample {

```

    public static void main(String[]
    args) {
        Animal.eat();
        Dog.eat();
    }
}

```

```

Animal is eating
Dog is eating

```

6.

Create a general class ThreeDObject and derive the classes Box, Cube, Cylinder and Cone from it. The class ThreeDObject has methods wholeSurfaceArea () and volume (). Override these two methods in each of the derived classes to calculate the volume and whole surface area of each type of three-dimensional objects. The dimensions of the objects are to be taken from the users and passed through the respective constructors of each derived class. Write a main method to test these classes.

Code: import

java.util.Scanner; class

```

ThreeDObject {
    public double
    wholeSurfaceArea() {return
    0.0;
    }
    public double
    volume() {return
    0.0;
    }
}

```

class Box extends

```
ThreeDObject {private
double length;
private double width;
private double height;
public Box(double length, double width, double
    height) {this.length = length;
    this.width = width;
    this.height = height;
}
public double wholeSurfaceArea() {
    return 2 * (length * width + width * height + height * length);
}
public double volume() {
    return length * width * height;
}
```

```
}  
class Cube extends  
    ThreeDObject {private double  
    side;  
    public Cube(double  
        side) {this.side = side;  
    }  
    public double  
        wholeSurfaceArea() {return 6  
        * side * side;  
    }  
    public double volume() {  
        return side * side *  
        side;  
    }  
}  
class Cylinder extends  
    ThreeDObject {private double  
    radius;  
    private double height;  
    public Cylinder(double radius, double  
        height) {this.radius = radius;  
        this.height = height;  
    }  
    public double wholeSurfaceArea() {  
        return 2 * Math.PI * radius * (radius + height);  
    }  
    public double volume() {  
        return Math.PI * radius * radius * height;  
    }  
}  
class Cone extends  
    ThreeDObject {private double  
    radius;  
    private double height;  
    public Cone(double radius, double  
        height) {this.radius = radius;  
        this.height = height;  
    }  
    public double wholeSurfaceArea() {  
        return Math.PI * radius * (radius + Math.sqrt(radius * radius + height * height));  
    }  
    public double volume() {  
        return (Math.PI * radius * radius * height) / 3.0;  
    }  
}  
public class TestThreeDObjects {
```

```
public static void main(String[]
args) {
    Scanner scanner = new
    Scanner(System.in);
    System.out.println("Enter dimensions for
    Box:");System.out.print("Length: ");
    double boxLength =
    scanner.nextDouble();
    System.out.print("Width: ");
    double boxWidth =
    scanner.nextDouble();
    System.out.print("Height: ");
    double boxHeight = scanner.nextDouble();
```

```
Box box = new Box(boxLength, boxWidth,
boxHeight);System.out.println("\nEnter
dimensions for Cube:"); System.out.print("Side:
");
double cubeSide =
scanner.nextDouble();Cube cube =
new Cube(cubeSide);
System.out.println("\nEnter dimensions for
Cylinder:");System.out.print("Radius: ");
double cylinderRadius =
scanner.nextDouble();
System.out.print("Height: ");
double cylinderHeight = scanner.nextDouble();
Cylinder cylinder = new Cylinder(cylinderRadius,
cylinderHeight);System.out.println("\nEnter dimensions for
Cone:"); System.out.print("Radius: ");
double coneRadius =
scanner.nextDouble();
System.out.print("Height: ");
double coneHeight = scanner.nextDouble();
Cone cone = new Cone(coneRadius,
coneHeight);System.out.println("\nResults:");
System.out.println("Box Surface Area: " + box.wholeSurfaceArea());
System.out.println("Box Volume: " + box.volume());
System.out.println("\nCube Surface Area: " +
cube.wholeSurfaceArea()); System.out.println("Cube Volume: " +
cube.volume()); System.out.println("\nCylinder Surface Area: " +
cylinder.wholeSurfaceArea());System.out.println("Cylinder Volume: " +
cylinder.volume()); System.out.println("\nCone Surface Area: " +
cone.wholeSurfaceArea()); System.out.println("Cone Volume: " +
cone.volume());
scanner.close();
}
```


}

```
Length: 10  
Width: 20  
Height: 30
```

```
Enter dimensions for Cube:  
Side: 40
```

```
Enter dimensions for Cylinder:  
Radius: 50  
Height: 60
```

```
Enter dimensions for Cone:  
Radius: 70  
Height: 20
```

```
Results:  
Box Surface Area: 2200.0  
Box Volume: 6000.0
```

```
Cube Surface Area: 9600.0  
Cube Volume: 64000.0
```

7. Write a program to create a class named Vehicle having protected instance variables regnNumber, speed, color, ownerName and a method showData () to show "This is a vehicleclass". Inherit the Vehicle class into subclasses named Bus and Car having individual private instance variables routeNumber in Bus and manufacturerName in Car and both of them having showData () method showing all details of Bus and Car respectively with content of the super class's showData () method.

```
Code: public class TestVehicle {
}
class Vehicle {
    protected String
    regnNumber;protected int
    speed; protected String
    color; protected String
    ownerName;
    public Vehicle(String regnNumber, int speed, String color, String
        ownerName) {this.regnNumber = regnNumber;
        this.speed =
        speed;this.color =
        color;
        this.ownerName = ownerName;
    }
    public void showData() {
        System.out.println("This is a vehicle
        class");
        System.out.println("Registration Number: " +
        regnNumber);System.out.println("Speed: " + speed);
        System.out.println("Color: " + color);
        System.out.println("Owner Name: " + ownerName);
    }
}
class Bus extends Vehicle
{ private String
    routeNumber;
    public Bus(String regnNumber, int speed, String color, String ownerName, String
        routeNumber) {super(regnNumber, speed, color, ownerName);
        this.routeNumber = routeNumber;
    }
    public void showData() {
        super.showData();
        System.out.println("This is a
        Bus");
        System.out.println("Route Number: " + routeNumber);
    }
}
class Car extends Vehicle {
    private String manufacturerName;
    public Car(String regnNumber, int speed, String color, String ownerName,
StringmanufacturerName) {
        super(regnNumber, speed, color,
        ownerName);this.manufacturerName =
        manufacturerName;
    }
    public void showData() {
```

```
super.showData();
System.out.println("This is a
Car");
System.out.println("Manufacturer Name: " + manufacturerName);
}
}
public class Test {
    public static void main(String[] args) {
        Bus myBus = new Bus("123", 60, "Blue", "John Doe",
"Route 1");
        Car myCar = new Car("456", 80, "Red", "Jane
Doe", "Toyota");
        System.out.println("Details of Bus:");
    }
}
```

```

myBus.showData();
System.out.println("\nDetails of
Car:");myCar.showData();
}
}

```

```

Details of Bus:
This is a vehicle class
Registration Number: 123
Speed: 60
Color: Blue
Owner Name: John Doe
This is a Bus
Route Number: Route 1

```

```

Details of Car:
This is a vehicle class
Registration Number: 456
Speed: 80
Color: Red
Owner Name: Jane Doe
This is a Car
Manufacturer Name: Toyota

```

8. An educational institution maintains a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown below. Write all the classes and define the methods to create the database and retrieve individual information as and when needed. Write a driver program to test the classes. Staff (code, name) Officer (grade) is a Staff RegularTypist (remuneration) is a Typist Teacher (subject, publication) is a Staff Typist (speed) is a Staff CasualTypist (daily wages) is a Typist.

```

Code: class Staff {
    private int code;
    private String
    name;
    public Staff(int code, String
        name) {this.code = code;
        this.name = name;
    }
    public int
        getCode() {
            return code;
        }
    public String
        getName() {return
            name;
        }
    }
}
class Typist extends
    Staff {private int
    speed;

    public Typist(int code, String name, int

```

```
    speed) {super(code, name);  
    this.speed = speed;  
}  
public int  
    getSpeed() {  
        return speed;  
    }  
}  
class Officer extends  
    Staff {private String  
    grade;
```

```
public Officer(int code, String name, String
    grade) {super(code, name);
    this.grade = grade;
}
public String
    getGrade() {return
    grade;
}
}
class RegularTypist extends
    Typist {private double
    remuneration;

    public RegularTypist(int code, String name, int speed, double
        remuneration) {super(code, name, speed);
        this.remuneration = remuneration;
    }
    public double
        getRemuneration() {return
        remuneration;
    }
}
class Teacher extends
    Staff {private String
    subject; private String
    publication;

    public Teacher(int code, String name, String subject, String
        publication) {super(code, name);
        this.subject = subject;
        this.publication = publication;
    }
    public String
        getSubject() {return
        subject;
    }
    public String
        getPublication() {return
        publication;
    }
}
class CasualTypist extends
    Typist {private double
    dailyWages;
    public CasualTypist(int code, String name, int speed, double
        dailyWages) {super(code, name, speed);
        this.dailyWages = dailyWages;
    }
}
```

```
public double
    getDailyWages() {return
        dailyWages;
    }
}

public class EducationalInstitution {
    public static void main(String[]
args) {
        Officer officer = new Officer(101, "John Doe", "Grade A");
        RegularTypist regularTypist = new RegularTypist(201, "Jane Smith", 60, 5000.0);
        Teacher teacher = new Teacher(301, "Alice Johnson", "Math", "Research in
Education");CasualTypist casualTypist = new CasualTypist(401, "Bob Brown",
50, 100.0);
    }
```

```

System.out.println("Officer Information:");
System.out.println("Code: " + officer.getCode());
System.out.println("Name: " + officer.getName());
System.out.println("Grade: " + officer.getGrade());
System.out.println("\nRegular Typist Information:");
System.out.println("Code: " + regularTypist.getCode());
System.out.println("Name: " + regularTypist.getName());
System.out.println("Speed: " + regularTypist.getSpeed());
System.out.println("Remuneration: " +
regularTypist.getRemuneration());System.out.println("\nTeacher
Information:");
System.out.println("Code: " + teacher.getCode());
System.out.println("Name: " + teacher.getName());
System.out.println("Subject: " + teacher.getSubject());
System.out.println("Publication: " + teacher.getPublication());
System.out.println("\nCasual Typist Information:");
System.out.println("Code: " + casualTypist.getCode());
System.out.println("Name: " + casualTypist.getName());
System.out.println("Speed: " + casualTypist.getSpeed());
System.out.println("Daily Wages: " + casualTypist.getDailyWages());
}
}

```

```

Officer Information:
Code: 101
Name: John Doe
Grade: Grade A

Regular Typist Information:
Code: 201
Name: Jane Smith
Speed: 60
Remuneration: 5000.0

Teacher Information:
Code: 301
Name: Alice Johnson
Subject: Math
Publication: Research in Education

Casual Typist Information:
Code: 401
Name: Bob Brown
Speed: 50
Daily Wages: 100.0

```

9. Create a base class Building that stores the number of floors of a building, number of rooms and it's total footage. Create a derived class House that inherits Building and alsostores the number of bedrooms and bathrooms. Demonstrate the working of the classes.Code: class Building {

```

protected int
numberOfFloors; protected
int numberOfRooms;
protected double
totalFootage;
public Building(int numberOfFloors, int numberOfRooms, double
totalFootage) {this.numberOfFloors = numberOfFloors;
this.numberOfRooms =
numberOfRooms;this.totalFootage =

```



```
        totalFootage;
    }
    public void displayInfo() {
        System.out.println("Number of Floors: " +
            numberOfFloors); System.out.println("Number of
            Rooms: " + numberOfRooms); System.out.println("Total
            Footage: " + totalFootage + " sq.ft");
    }
}
class House extends Building {
    private int
    numberOfBedrooms;
```

```

private int numberOfBathrooms;
public House(int numberOfFloors, int numberOfRooms, double
totalFootage, int numberOfBedrooms, int numberOfBathrooms) {
    super(numberOfFloors, numberOfRooms,
    totalFootage); this.numberOfBedrooms =
    numberOfBedrooms; this.numberOfBathrooms =
    numberOfBathrooms;
}
public void
displayInfo() {
    super.displayInfo();
    System.out.println("Number of Bedrooms: " + numberOfBedrooms);
    System.out.println("Number of Bathrooms: " + numberOfBathrooms);
}
}
public class Main {
    public static void main(String[] args) {
        House myHouse = new House(2, 5, 2000.0,
        3, 2); System.out.println("House
        Information:"); myHouse.displayInfo();
    }
    House Information:
    Number of Floors: 2
    Number of Rooms: 5
    Total Footage: 2000.0 sq.ft
    Number of Bedrooms: 3
    Number of Bathrooms: 2
    PS C:\Users\User\Desktop\Java\practice>
}

```

10. In the earlier program, create a second derived class Office that inherits Building and stores the number of telephones and tables. Now demonstrate the working of all three classes.

Code:

```

class Building {
    protected String
    address; protected int
    floors;
    public Building(String address, int
    floors) { this.address = address;
    this.floors = floors;
    }
    public void display() {
        System.out.println("Address: " + address);
        System.out.println("Number of floors: " +
        floors);
    }
}

```

```
class Office extends
    Building {private int
    telephones; private int
    tables;
    public Office(String address, int floors, int telephones, int
        tables) {super(address, floors);
        this.telephones =
        telephones;this.tables =
        tables;
    }
    public void
        display() {
        super.display();
```

```

        System.out.println("Number of telephones: " +
        telephones);System.out.println("Number of tables: " +
        tables);
    }
}
class House extends
    Building {private int
    bedrooms; private int
    bathrooms;
    public House(String address, int floors, int bedrooms, int
        bathrooms) {super(address, floors);
        this.bedrooms =
        bedrooms; this.bathrooms
        = bathrooms;
    }
    public void
    display() {
        super.display();
        System.out.println("Number of bedrooms: " + bedrooms);
        System.out.println("Number of bathrooms: " + bathrooms);
    }
}
public class BuildingDemo {
    public static void main(String[] args) {
        Office office = new Office("123 Main St", 5, 20,
        50);House house = new House("456 Elm St",
        2, 3, 2); System.out.println("Office details:");
        office.display();
        System.out.println("\nHouse
        details:");house.display();
    }
}

```

```

Office details:
Address: 123 Main St
Number of floors: 5
Number of telephones: 20
Number of tables: 50

```

```

House details:
Address: 456 Elm St
Number of floors: 2
Number of bedrooms: 3
Number of bathrooms: 2

```

11. Write a Java program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class HexNum which inherits Num and overrides shownum() which

displays the hexadecimal value of the number. Demonstrate the working of the classes.

```
Code: class Num {  
    protected int number;  
    public Num(int  
    number) {  
        this.number = number;  
    }  
    public void shownum() {  
        System.out.println("Decimal Value: " +  
        number);  
    }  
}  
class HexNum extends Num {  
    public HexNum(int number) {
```

```

    super(number);
}
public void shownum() {
    System.out.println("Hexadecimal Value: " + Integer.toHexString(number));
}
}
public class NumDemo {
    public static void main(String[]
        args) {Num numObj = new
        Num(255);
        System.out.println("Using base class
        Num:");numObj.shownum();
        HexNum hexNumObj = new HexNum(255);
        System.out.println("\nUsing derived class
        HexNum:");hexNumObj.shownum();
    }
}

```

```

Using base class Num:
Decimal Value: 255

```

```

Using derived class HexNum:
Hexadecimal Value: ff

```

12. Write a Java program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class OctNum which inherits Num and overrides shownum() which displays the octal value of the number. Demonstrate the working of the classes.

Code:

```

class Num {
    protected int number;
    public Num(int
        number) {
        this.number = number;
    }
    public void showNum() {
        System.out.println("Number: " +
        number);
    }
}
class OctNum extends Num {
    public OctNum(int number) {
        super(number);
    }
    public void showNum() {
        System.out.println("Octal Value: " + Integer.toOctalString(number));
    }
}
public class Main {
    public static void main(String[]

```

```
args) {Num num = new  
Num(10);  
OctNum octNum = new OctNum(10);  
System.out.println("Num Class:");  
num.showNum();  
System.out.println("\nOctNum Class:");  
octNum.showNum();  
}  
}
```

Num Class:

Number: 15

PS C:\Users\User\Desktop\Java\practice>

13. Combine Question number 10 and 11 and have all the three classes together. Now describe the working of all classes.

Code: class Num

```
{int number;
public Num(int number) {
    this.number = number;
}
public void shownum() {
    System.out.println("Decimal: " +
        number);
}
}
class HexNum extends Num {
    public HexNum(int number) {
        super(number);
    }
    public void shownum() {
        System.out.println("Hexadecimal: " + Integer.toHexString(number));
    }
}
class OctNum extends Num {
    public OctNum(int number) {
        super(number);
    }
    public void shownum() {
        System.out.println("Octal: " + Integer.toOctalString(number));
    }
}
public class Main {
    public static void main(String[]
        args) {Num num1 = new
        Num(10); num1.shownum();
        HexNum hex_num = new
        HexNum(255);hex_num.shownum();
        OctNum oct_num = new
        OctNum(64);oct_num.shownum();
    }
}
```

```
Decimal: 10
Hexadecimal: ff
Octal: 100
```

14. Create a base class Distance which stores the distance between two locations in miles and a method travelTime(). The method prints the time taken to cover the

distance when the speed is 60 miles per hour. Now in a derived class **DistanceMKS**, override `travelTime()` so that it prints the time assuming the distance is in kilometers and the speed is 100 km per second. Demonstrate the working of the classes.

Code: class Distance {

```

protected double distanceMiles;
public Distance(double
    distanceMiles) {this.distanceMiles
    = distanceMiles;
}
public void travelTime() {
    double speedMilesPerHour = 60.0;
    double timeHours = distanceMiles / speedMilesPerHour;
    System.out.println("Time taken to cover " + distanceMiles + " miles at 60 miles per
hour: " +timeHours + " hours");
}
}
class DistanceMKS extends Distance {
    public DistanceMKS(double
        distanceMiles) {super(distanceMiles);
    }
    public void travelTime() {
        double speedKilometersPerSecond = 100.0 / 3600.0; // converting speed from km per
secondto km per hour
        double distanceKilometers = distanceMiles * 1.60934; // converting distance from
miles tokilometers
        double timeSeconds = distanceKilometers / speedKilometersPerSecond;
        System.out.println("Time taken to cover " + distanceMiles + " miles at 100 km per
second: " +
timeSeconds + " seconds");
    }
}
}
public class DistanceDemo {
    public static void main(String[] args) {
        Distance distanceObj = new
        Distance(100);
        System.out.println("Using base class
Distance:");distanceObj.travelTime();
        DistanceMKS distanceMKSObj = new
        DistanceMKS(100);        System.out.println("\nUsing
derived        class        DistanceMKS:");
        distanceMKSObj.travelTime();
    }
}

```

```

Using base class Distance:
Time taken to cover 100.0 miles at 60 miles per hour: 1.6666666666666667 hours

```

```

Using derived class DistanceMKS:
Time taken to cover 100.0 miles at 100 km per second: 5793.624 seconds

```

15. Create a base class called “vehicle” that stores number of wheels and speed. Create thefollowing derived classes – “car” that inherits “vehicle” and also stores number of passengers. “truck” that inherits “vehicle” and also stores the load limit. Write a main function to create objects of these two derived classes and display all the information about“car” and “truck”. Also compare the speed of these two vehicles - car and truck and displaywhich one is faster.

```

Code: class Vehicle {
    protected int
}

```

```
numberOfWheels;protected
int speed;
public Vehicle(int numberOfWheels, int
    speed) {this.numberOfWheels =
    numberOfWheels; this.speed = speed;

public int
    getSpeed() {
        return speed;
```

```
}
```

```
public void displayInfo() {
    System.out.println("Number of Wheels: " +
        numberOfWheels);System.out.println("Speed: " + speed
        + " km/h");
}
}
class Car extends Vehicle {
    private int numberOfPassengers;
    public Car(int numberOfWheels, int speed, int
        numberOfPassengers) {super(numberOfWheels, speed);
        this.numberOfPassengers = numberOfPassengers;
    }
    public void
        displayInfo() {
            super.displayInfo();
            System.out.println("Number of Passengers: " + numberOfPassengers);
        }
}
class Truck extends
    Vehicle {private int
        loadLimit;
    public Truck(int numberOfWheels, int speed, int
        loadLimit) {super(numberOfWheels, speed);
        this.loadLimit = loadLimit;
    }
    public void
        displayInfo() {
            super.displayInfo();
            System.out.println("Load Limit: " + loadLimit + " tons");
        }
}
public class Main {
    public static void main(String[] args) {
        Car myCar = new Car(4, 120, 5);
        Truck myTruck = new Truck(6, 80,
            10);System.out.println("Car
            Information:"); myCar.displayInfo();
        System.out.println("\nTruck
            Information:");myTruck.displayInfo();
        if (myCar.getSpeed() > myTruck.getSpeed()) {
            System.out.println("\nCar is faster than
                Truck.");
        } else if (myCar.getSpeed() <
            myTruck.getSpeed()) {
            System.out.println("\nTruck is faster than
                Car.");
        }
    }
}
```

```
} else {  
    System.out.println("\nCar and Truck have the same speed.");  
}  
}
```

```
Car Information:  
Number of Wheels: 4  
Speed: 120 km/h  
Number of Passengers: 5
```

```
Truck Information:  
Number of Wheels: 6  
Speed: 80 km/h  
Load Limit: 10 tons
```

```
Car is faster than Truck.
```

```
}
```

16. Write a Java program to explain “multilevel inheritance.”**Code:**

```
class Animal {  
    void eat() {  
        System.out.println("Animal is eating.");  
    }  
}  
  
class Dog extends  
    Animal {  
    void bark() {  
        System.out.println("Dog is barking.");  
    }  
    Dog() {  
        super();  
    }  
}  
  
class Bulldog extends  
    Dog {  
    void guard() {  
        System.out.println("Bulldog is guarding.");  
    }  
    Bulldog()  
    {  
        super()  
        ;  
    }  
}  
  
public class Main {  
    public static void main(String[]  
        args) {  
        Bulldog myDog = new  
            Bulldog();  
        myDog.eat();  
        myDog.bark();  
        myDog.guard(  
            );  
    }  
}  
  
Animal is eating.  
Dog is barking.  
Bulldog is guarding.  
PS C:\Users\User\Desktop\Java\practice>
```