DEPARTMENT OF INFORMATION TECHNOLOGY

22IT409: LAB. OBJECT ORIENTED PROGRAMMING

LAB MANUAL

1. Introduction to Java as an Object Oriented Programming Language, with sample program.

Program:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Output:

```
Hello, World!
```

2. (A) Write a java program to give the demonstration how to use of data types and control structures.

You have just started a sales job in a department store. Your pay consists of a base salary and a commission. The base salary is $5,000. The scheme shown below is used to determine the commission rate.

Sales Amount	Commission Rate
$0.01–$5,000	8 percent
$5,000.01–$10,000	10 percent
$10,000.01 and above	12 percent

Your goal is to earn $30,000 in a year. Write a program that will find out the minimum amount of sales you have to generate in order to make $30,000.

Program:

```
public class SalesGoalCalculator {
    public static void main(String[] args) {
        int baseSalary = 5000;
        double goal = 30000;

        float sales = findMinimumSalesAmount(baseSalary, goal);

        System.out.println("Minimum sales amount needed to earn $30,000: $" +
sales);
    }

    public static float findMinimumSalesAmount(int baseSalary, double goal) {
        float sales = 0;
        double commission;
        while (true) {
            if (sales <= 5000) {
                commission = sales * 0.08;
            } else if (sales <= 10000) {
                commission = 5000 * 0.08 + (sales - 5000) * 0.10;
            } else {
```

```
                commission = 5000 * 0.08 + 5000 * 0.10 + (sales - 10000) * 0.12;
            }

            if (baseSalary + commission >= goal) {
                break;
            }

            sales += 0.01;
        }

        return sales;
    }
}
```

Output:

```
Minimum sales amount needed to earn $30,000: $210833.34
```

2 (B) Your program prints five lines. Each line consists of three parts. The first part comprises the spaces before the numbers; the second part, the leading numbers, such as 3 2 1 on line 3; and the last part, the ending numbers, such as 2 3 on line 3

Program:

```
public class NumberPattern {

    public static void main(String[] args) {

        int rows = 5;


        for (int i = 1; i <= rows; i++) {

            for (int j = rows - i; j > 0; j--) {

                System.out.print("  ");

            }


            for (int j = i; j >= 1; j--) {

                System.out.print(j + " ");

            }


            for (int j = 2; j <= i; j++) {

                System.out.print(j + " ");

            }


            System.out.println();

        }

    }

}
```

Output:

```
    1
   212
  32123
 4321234
543212345
```

2 (C) Displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is prime if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Program:

```
public class FirstFiftyPrimes {
    public static void main(String[] args) {
        int count = 0;
        int number = 2;
        System.out.println("The first 50 prime numbers are:");

        while (count < 50) {
            if (isPrime(number)) {
                System.out.print(number + " ");
                count++;

                if (count % 10 == 0) {
                    System.out.println();
                }
            }
            number++;
        }
    }

    public static boolean isPrime(int n) {
        if (n <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

Output:

```
The first 50 prime numbers are:

2 3 5 7 11 13 17 19 23 29

31 37 41 43 47 53 59 61 67 71

73 79 83 89 97 101 103 107 109 113

127 131 137 139 149 151 157 163 167 173

179 181 191 193 197 199 211 223 227 229
```

3 (A) (i) Write a Java method to compute the future investment value at a given interest rate for a specified number of years.

Program:

```
import java.util.Scanner;

public class FutureInvestmentValue {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);


        System.out.print("Input the investment amount: ");

        double investmentAmount = input.nextDouble();


        System.out.print("Input the rate of interest: ");

        double interestRate = input.nextDouble() / 100;


        System.out.print("Input number of years: ");

        int years = input.nextInt();


        System.out.println("Years\t\t\tFuture Value");


        for (int i = 1; i <= years; i++) {

            double futureValue = investmentAmount * Math.pow(1 + (interestRate /
12), i * 12);

            System.out.println(i + "\t\t\t" + String.format("%.2f", futureValue));

        }

        input.close();

    }

}
```

Output:

```
Input the investment amount: 1000

Input the rate of interest: 10

Input number of years: 5

Years           Future Value

1               1104.71

2               1220.39

3               1348.18

4               1489.35

5               1645.31
```

3 (A) (ii) Write a Java method to count the number of digits in an integer with the value 2. The integer may be assumed to be nonnegative.

Program:

```
import java.util.Scanner;


public class DigitCounter {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Input a number: ");
        int number = input.nextInt();
        System.out.println("Frequency of digit 2: " + countDigit2(number));
        input.close();
    }


    public static int countDigit2(int number) {
        int count = 0;
        while (number > 0) {
            if (number % 10 == 2) {
                count++;
            }
            number /= 10;
        }
        return count;
    }
}
```

Output:

```
Input a number: 34222

Frequency of digit 2: 3
```

3 (B) Print Fibonacci series upto 10 numbers (use recursive method)

Program:

```
import java.util.Scanner;

public class Fibonacci {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of terms in Fibonacci series: ");
        int n = scanner.nextInt();
        scanner.close();

        System.out.println("Fibonacci series upto " + n + " numbers:");
        for (int i = 0; i < n; i++) {
            System.out.print(fibonacci(i) + " ");
        }
    }

    public static int fibonacci(int n) {
        if (n == 1 || n == 0) {
            return n;
        }
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

Output:

```
Enter the number of terms in Fibonacci series: 10

Fibonacci series upto 10 numbers:

0 1 1 2 3 5 8 13 21 34
```

4 (A) Print the sum, difference and product of two complex numbers by creating a class named 'MyComplex' with separate methods for each operation whose real and imaginary parts are entered by user.

Program:

```
import java.util.Scanner;

class MyComplex {
    private double real;
    private double imag;

    public MyComplex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }

    public void display() {
        if (this.imag >= 0) {
            System.out.println(this.real + " + " + this.imag + "i");
        } else {
            System.out.println(this.real + " " + this.imag + "i");
        }
    }

    public MyComplex add(MyComplex num) {
        double realSum = this.real + num.real;
        double imagSum = this.imag + num.imag;
        return new MyComplex(realSum, imagSum);
    }

    public MyComplex subtract(MyComplex num) {
        double realDiff = this.real - num.real;
        double imagDiff = this.imag - num.imag;
        return new MyComplex(realDiff, imagDiff);
    }
```

```
    public MyComplex multiply(MyComplex num) {
        double realProduct = this.real * num.real - this.imag * num.imag;
        double imagProduct = this.real * num.imag + this.imag * num.real;
        return new MyComplex(realProduct, imagProduct);
    }
}

public class ComplexOperations {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter real part of first complex number:");
        double real1 = scanner.nextDouble();
        System.out.print("Enter imaginary part of first complex number:");
        double imaginary1 = scanner.nextDouble();

        System.out.print("\nEnter real part of second complex number:");
        double real2 = scanner.nextDouble();
        System.out.print("Enter imaginary part of second complex number:");
        double imaginary2 = scanner.nextDouble();

        MyComplex complex1 = new MyComplex(real1, imaginary1);
        MyComplex complex2 = new MyComplex(real2, imaginary2);

        System.out.println("\nFirst Number: ");
        complex1.display();

        System.out.println("\nSecond Number: ");
        complex2.display();
        MyComplex sum = complex1.add(complex2);
        System.out.print("\nSum: ");
        sum.display();
        MyComplex difference = complex1.subtract(complex2);
        System.out.print("\nDifference: ");
        difference.display();
```

```
         MyComplex product = complex1.multiply(complex2);
        System.out.print("\nProduct: ");
        product.display();
        scanner.close();
    }
}
```

Output:

```
Enter real part of first complex number:7
Enter imaginary part of first complex number:8

Enter real part of second complex number:9
Enter imaginary part of second complex number:1

First Number:
7.0 + 8.0i

Second Number:
9.0 + 1.0i

Sum: 16.0 + 9.0i

Difference: -2.0 + 7.0i

Product: 55.0 + 79.0i
```

4 (B) Create a class InchFeet with method to display totalDistance after adding two distances also write a method to find the difference between two distances.

Program:

```
import java.util.Scanner;


public class InchFeet {
    private int inches;
    private int feet;


    public InchFeet(int feet, int inches) {
        this.feet = feet;
        this.inches = inches;
    }


    public void displayDistance(int n) {
        System.out.println("\nDistance"+ n + " : " + feet + " feet " + inches + "
inches");
    }


    public void displayTotalDistance(InchFeet other) {
        int totalInches = this.inches + other.inches;
        int totalFeet = this.feet + other.feet + (totalInches / 12);
        totalInches %= 12;
        System.out.println("\nTotal Distance: " + totalFeet + " feet " +
totalInches + " inches");
    }


    public void findDifference(InchFeet other) {
        int thisInches = this.feet * 12 + this.inches;
        int otherInches = other.feet * 12 + other.inches;
        int difference = Math.abs(thisInches - otherInches);
        int diffFeet = difference / 12;
        int diffInches = difference % 12;
        System.out.println("\nDifference: " + diffFeet + " feet " + diffInches + "
inches");
```

```
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter distance 1 in feet: ");
        int feet1 = scanner.nextInt();
        System.out.print("Enter distance 1 in inches: ");
        int inches1 = scanner.nextInt();

        System.out.print("\nEnter distance 2 in feet: ");
        int feet2 = scanner.nextInt();
        System.out.print("Enter distance 2 in inches: ");
        int inches2 = scanner.nextInt();

        InchFeet distance1 = new InchFeet(feet1, inches1);
        InchFeet distance2 = new InchFeet(feet2, inches2);

        distance1.displayDistance(1);
        distance2.displayDistance(2);

        distance1.displayTotalDistance(distance2);
        distance1.findDifference(distance2);

        scanner.close();
    }
}
```

Output:

```
Enter distance 1 in feet: 5

Enter distance 1 in inches: 3


Enter distance 2 in feet: 5

Enter distance 2 in inches: 7


Distance1 : 5 feet 3 inches


Distance2 : 5 feet 7 inches


Total Distance: 10 feet 10 inches


Difference: 0 feet 4 inches
```

5 (A) Write a class called Room, which has three private instance variables: - a double width, representing the width of the room in feet, - a double length, representing the length of the room in feet, and - an int floor, representing the building floor that the room is on.

a) Write a default constructor for the class Room that sets the width to 10, the length to 12.5, and the floor to 1.

b) Write get and set methods ("getters" and "setters") for the three instance variables. For the set methods for the width and length, only positive values should be set. If the input is 0 or a negative number, the variables should not be changed.

c) Write a constructor for the class Room that takes in two double parameters and an int, and sets length to the larger double, width to the smaller double, and floor to the int. Use the setters from part (b).

d) Override the default toString method for the class Room, so that it a String with the form "length x width on floor floor". For example, calling toString on the instance of Room created by the default constructor from part (a) would return the String "12.5 x 10, floor 1".

e) Implement the Comparable interface for the class Room. The method CompareTo should be written so that an array of Rooms will be sorted first by floor, then by length, and finally by width. Test this method by creating an arrays of Rooms, and sorting them using Arrays.sort(...).

f) Write a subclass of the class Room called Classroom, which also has a private instance variable of type int called numStudents, representing the maximum number of students that the classroom can hold.

g) Write a constructor for Classroom which takes in two double variables and two ints. The instance variable length should be set to the larger double, and the width should be set to the smaller double as in Room. The instance variable floor should be set to the first int, and the instance variable numStudents should be set to the second int. Leave the instance variables as private in Room and use setters to access them.

h) Write a toString method for Classroom which uses the toString method for Room, followed by an additional String ", capacity = numStudents students", where numStudents is replaced by the instance variable value.

i) Write a static method which takes in an array of Rooms and returns an array of Classrooms that contains exactly those Rooms in the input array that are also Classrooms.

Program:

```
import java.util.Arrays;


public class Room implements Comparable<Room> {

    private double width;

    private double length;

    private int floor;


    public Room() {

        this.width = 10;

        this.length = 12.5;
```

```
        this.floor = 1;
    }

    public Room(double width, double length, int floor) {
        setWidth(width);
        setLength(length);
        setFloor(floor);
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        if (width > 0) {
            this.width = width;
        }
    }

    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        if (length > 0) {
            this.length = length;
        }
    }

    public int getFloor() {
        return floor;
    }

    public void setFloor(int floor) {
        this.floor = floor;
    }
```

```
    public String toString() {
        return String.format("Room: %.1f ft x %.1f ft on floor %d", length, width,
floor);
    }

    public int compareTo(Room otherRoom) {
        int floorComparison = Integer.compare(this.floor, otherRoom.floor);

        if (floorComparison != 0) {
            return floorComparison;
        }

        int lengthComparison = Double.compare(this.length, otherRoom.length);

        if (lengthComparison != 0) {
            return lengthComparison;
        }

        return Double.compare(this.width, otherRoom.width);
    }

    public static void main(String[] args) {
        Room room1 = new Room();
        Room room2 = new Room(8.5, 11, 2);

        System.out.println("Room 1 details: \n" + room1 + "\n");
        System.out.println("Room 2 details: \n" + room2 + "\n");

        int comparisonResult = room1.compareTo(room2);
        if (comparisonResult < 0) {
            System.out.println("Room 1 is smaller in size and/or at a lower floor
level than Room 2");
        } else if (comparisonResult > 0) {
            System.out.println("Room 1 is larger in size and/or at a higher floor
level than Room 2");
        } else {
```

```
            System.out.println("Room 1 is equal in size and floor level to Room
2");
        }

        Room[] rooms = {room2, room1};
        System.out.println("\nBefore sorting:");
        for (Room room : rooms) {
            System.out.println(room);
        }

        Arrays.sort(rooms);

        System.out.println("\nAfter sorting:");
        for (Room room : rooms) {
            System.out.println(room);
        }
    }
}

class Classroom extends Room {
    private int numStudents;

    public Classroom(double width, double length, int floor, int numStudents) {
        super(width, length, floor);
        setNumStudents(numStudents);
    }

    public int getNumStudents() {
        return numStudents;
    }

    public void setNumStudents(int numStudents) {
        this.numStudents = numStudents;
    }

    public String toString() {
```

```
        return super.toString() + String.format(", capacity = %d students",
numStudents);
    }

    public static Classroom[] filterClassrooms(Room[] rooms) {
        int numClassrooms = 0;
        for (Room room : rooms) {
            if (room instanceof Classroom) {
                numClassrooms++;
            }
        }

        Classroom[] classrooms = new Classroom[numClassrooms];

        int index = 0;
        for (Room room : rooms) {
            if (room instanceof Classroom) {
                classrooms[index] = (Classroom) room;
                index++;
            }
        }

        return classrooms;
    }
}
```

Output:

```
Room 1 details:
Room: 12.5 ft x 10.0 ft on floor 1

Room 2 details:
Room: 11.0 ft x 8.5 ft on floor 2

Room 1 is smaller in size and/or at a lower floor level than Room 2

Before sorting:
Room: 11.0 ft x 8.5 ft on floor 2
Room: 12.5 ft x 10.0 ft on floor 1

After sorting:
Room: 12.5 ft x 10.0 ft on floor 1
Room: 11.0 ft x 8.5 ft on floor 2
```

5 (B) Write a program to get details of publisher (publisher-name, author –name) and book (book-title, book-price) .Display the details of publisher as well as book using function overriding.

Program:

```
import java.util.Scanner;

class Publisher {
    private String publisherName;
    private String authorName;

    public Publisher(String publisherName, String authorName) {
        this.publisherName = publisherName;
        this.authorName = authorName;
    }

    public void display() {
        System.out.println("Publisher Name: " + publisherName);
        System.out.println("Author Name: " + authorName);
    }
}

class Book extends Publisher {
    private String bookTitle;
    private double bookPrice;

    public Book(String publisherName, String authorName, String bookTitle, double
bookPrice) {
        super(publisherName, authorName);
        this.bookTitle = bookTitle;
        this.bookPrice = bookPrice;
    }

    public void display() {
        super.display();
        System.out.println("Book Title: " + bookTitle);
```

```
        System.out.println("Book Price: $" + bookPrice);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of books: ");
        int n = scanner.nextInt();

        for (int i = 1; i <= n; i++) {
            scanner.nextLine();
            System.out.println("\nEnter details for Book " + i + ":");

            System.out.print("Publisher Name: ");
            String publisherName = scanner.nextLine();

            System.out.print("Author Name: ");
            String authorName = scanner.nextLine();

            System.out.print("Book Title: ");
            String bookTitle = scanner.nextLine();

            System.out.print("Book Price: ");
            double bookPrice = scanner.nextDouble();

            Book book = new Book(publisherName, authorName, bookTitle, bookPrice);

            System.out.println("\nPublisher and Book Details for Book " + i +
":");
            book.display();
       }
        scanner.close();
    }
}
```

Output:

```
Enter the number of books: 2


Enter details for Book 1:
Publisher Name: RK Publications
Author Name: Raunak
Book Title: World
Book Price: 700

Publisher and Book Details for Book 1:
Publisher Name: RK Publications
Author Name: Raunak
Book Title: World
Book Price: $700.0

Enter details for Book 2:
Publisher Name: PG Prints
Author Name: Prushthi
Book Title: Beauty
Book Price: 300

Publisher and Book Details for Book 2:
Publisher Name: PG Prints
Author Name: Prushthi
Book Title: Beauty
Book Price: $300.0
```

6. Write a program to create interface name bank transaction (deposit() and withdraw()) depending upon the type of account (savings with zero and current with minimum Rs.2000 balance). Create one new class called Rate of Interest (ROI) and to use method calculate Interest.

Program:

```
import java.util.Scanner;

abstract class Account {
    protected double balance;

    public Account(double initialBalance) {
        if (initialBalance < 0) {
            System.out.print("Initial balance cannot be negative.");
        }
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        if (amount <= 0) {
            System.out.println("Invalid amount for deposit.");
            return;
        }
        balance += amount;
        System.out.println("Deposited: " + amount);
    }

    public abstract void withdraw(double amount);

    public double getBalance() {
        return balance;
    }
}

class SavingsAccount extends Account {
    public SavingsAccount(double initialBalance) {
```

```
        super(initialBalance);
    }

    public void withdraw(double amount) {
        if (amount <= 0 || balance < amount) {
            System.out.println("Insufficient balance or invalid amount for
withdrawal.");
            return;
        }
        balance -= amount;
        System.out.println("Withdrawn: " + amount);
    }
}

class CurrentAccount extends Account {
    public CurrentAccount(double initialBalance) {
        super(initialBalance);
    }

    public void withdraw(double amount) {
        if (amount <= 0) {
            System.out.println("Invalid amount for withdrawal.");
            return;
        }
        if (balance - amount < 0) {
            System.out.println("Insufficient balance for withdrawal. Overdraft not
allowed.");
            return;
        }
        balance -= amount;
        System.out.println("Withdrawn: " + amount);
    }
}

class ROI {
    public static double calculateInterest(double principal, double rate, int
time) {
```

```
        if (principal <= 0 || rate <= 0 || time <= 0) {
            System.out.print("Principal, rate, and time must be positive.");
        }
        return (principal * rate * time) / 100;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter initial balance for savings account: ");
        double initialSavingsBalance = scanner.nextDouble();
        SavingsAccount savingsAccount = new SavingsAccount(initialSavingsBalance);

        System.out.print("Enter initial balance for current account: ");
        double initialCurrentBalance = scanner.nextDouble();
        CurrentAccount currentAccount = new CurrentAccount(initialCurrentBalance);

        System.out.print("\nEnter deposit amount for savings account: ");
        double depositSavings = scanner.nextDouble();
        savingsAccount.deposit(depositSavings);

        System.out.print("Enter withdrawal amount for savings account: ");
        double withdrawSavings = scanner.nextDouble();
        savingsAccount.withdraw(withdrawSavings);
        System.out.println("Savings Account Balance: " +
savingsAccount.getBalance());

        System.out.print("\nEnter deposit amount for current account: ");
        double depositCurrent = scanner.nextDouble();
        currentAccount.deposit(depositCurrent);

        System.out.print("Enter withdrawal amount for current account: ");
        double withdrawCurrent = scanner.nextDouble();
        currentAccount.withdraw(withdrawCurrent);
```

```
        System.out.println("Current Account Balance: " +
currentAccount.getBalance());


        System.out.print("\nEnter principal amount for interest calculation: ");
        double principal = scanner.nextDouble();
        System.out.print("Enter interest rate: ");
        double rate = scanner.nextDouble();
        System.out.print("Enter time (in years) for interest calculation: ");
        int time = scanner.nextInt();
        double interest = ROI.calculateInterest(principal, rate, time);
        System.out.println("Interest calculated: " + interest);


        scanner.close();
    }
}
```

Output:

```
Enter initial balance for savings account: 1000
Enter initial balance for current account: 1000

Enter deposit amount for savings account: 4000
Deposited: 4000.0
Enter withdrawal amount for savings account: 300
Withdrawn: 300.0
Savings Account Balance: 4700.0

Enter deposit amount for current account: 3400
Deposited: 3400.0
Enter withdrawal amount for current account: 400
Withdrawn: 400.0
Current Account Balance: 4000.0

Enter principal amount for interest calculation: 10000
Enter interest rate: 10
Enter time (in years) for interest calculation: 5
Interest calculated: 5000.0
```

7. Create a City class that contains the list of the city with its zip code. It contains findCityByZipCode (int zipCode) method. If it doesn't find city name, then it'll throw the exception "City not found". Use the exception in program.

Program:

```
import java.util.Scanner;
class CityNotFoundException extends Exception {
    public CityNotFoundException(String message) {
        super(message);
    }
}

class City {
    private String[] cityNames;
    private int[] zipCodes;
    private int size;

    public City(int capacity) {
        cityNames = new String[capacity];
        zipCodes = new int[capacity];
        size = 0;
    }

    public void addCity(String cityName, int zipCode) {
        cityNames[size] = cityName;
        zipCodes[size] = zipCode;
        size++;
    }

    public String findCityByZipCode(int zipCode) throws CityNotFoundException {
        for (int i = 0; i < size; i++) {
            if (zipCodes[i] == zipCode) {
                return cityNames[i];
            }
        }
```

```
        throw new CityNotFoundException("City not found for zip code: " +
zipCode);
    }
}


public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of cities: ");
        int numOfCities = scanner.nextInt();
        scanner.nextLine();

        City city = new City(numOfCities);

        for (int i = 0; i < numOfCities; i++) {
            System.out.println("\nDetails of city "+(i+1)+":");
            System.out.print("Enter city name: ");
            String cityName = scanner.nextLine();
            System.out.print("Enter zip code: ");
            int zipCode = scanner.nextInt();
            scanner.nextLine();
            System.out.println();
            city.addCity(cityName, zipCode);
        }

        System.out.print("Enter zip code to find city: ");
        int searchZipCode = scanner.nextInt();
        scanner.nextLine();

        try {
            String foundCity = city.findCityByZipCode(searchZipCode);
            System.out.println("City found: " + foundCity);
        } catch (CityNotFoundException e) {
            System.out.println(e.getMessage());
        }
```

```
        scanner.close();
    }
}
```

Output:

```
Enter the number of cities: 2

Details of city 1:
Enter city name: Nagpur
Enter zip code: 1

Details of city 2:
Enter city name: Pune
Enter zip code: 2

Enter zip code to find city: 2
City found: Pune
```

8. Write a Java method to check whether a string is a valid password. Password rules:

A password must have at least ten characters.

A password consists of only letters and digits.

A password must contain at least two digits

Program:

```
import java.util.Scanner;


public class PasswordValidator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Input a password (You are agreeing to the below Terms and Conditions.):\n" +
                "1. A password must have at least ten characters.\n" +
                "2. A password consists of only letters and digits.\n" +
                "3. A password must contain at least two digits.");

        String password = scanner.nextLine();

        if (isValidPassword(password)) {
            System.out.println("Password is valid: " + password);
        } else {
            System.out.println("Password is invalid.");
        }
    }

    public static boolean isValidPassword(String password) {
        return hasRequiredLength(password) &&
                consistsOfLettersAndDigits(password) &&
                hasRequiredDigits(password);
    }

    private static boolean hasRequiredLength(String password) {
        return password.length() >= 10;
    }
```

```
    private static boolean consistsOfLettersAndDigits(String password) {
        for (char c : password.toCharArray()) {
            if (!Character.isLetterOrDigit(c)) {
                return false;
            }
        }
        return true;
    }

    private static boolean hasRequiredDigits(String password) {
        int digitCount = 0;
        for (char c : password.toCharArray()) {
            if (Character.isDigit(c)) {
                digitCount++;
            }
        }
        return digitCount >= 2;
    }
}
```

Output:

```
Input a password (You are agreeing to the below Terms and Conditions.):
1. A password must have at least ten characters.
2. A password consists of only letters and digits.
3. A password must contain at least two digits.
ycceit2024
Password is valid: ycceit2024
```

9. Write a program to create a java linked list. Methods of java linked list to use Add elements, Access element, Change elements, Remove elements.

Program:

```
import java.util.LinkedList;
import java.util.Scanner;

public class LinkedList {
    public static void main(String[] args) {
        LinkedList<String> linkedList = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter number of elements to add: ");
        int numElements = scanner.nextInt();
        scanner.nextLine();

        for (int i = 0; i < numElements; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            String element = scanner.nextLine();
            linkedList.add(element);
        }

        System.out.println("Linked List: " + linkedList);
        System.out.println("First element: " + linkedList.getFirst());
        System.out.println("Last element: " + linkedList.getLast());

        if (!linkedList.isEmpty()) {
            System.out.println("Do you want to change an element? (0/1)");
            String changeChoice = scanner.nextLine();
            if (changeChoice.equalsIgnoreCase("1")) {
                System.out.print("Enter the index of the element to change: ");
                int index = scanner.nextInt();
                scanner.nextLine();
                if (index >= 0 && index < linkedList.size()) {
                    System.out.print("Enter the new element: ");
```

```
                    String newElement = scanner.nextLine();
                    linkedList.set(index, newElement);
                    System.out.println("Linked List after changing: " +
linkedList);
                }

                else {
                    System.out.println("Invalid index!");
                }
            }
        }

        if (!linkedList.isEmpty()) {
            System.out.println("Do you want to remove an element? (0/1)");
            String removeChoice = scanner.nextLine();
            if (removeChoice.equalsIgnoreCase("1")) {
                System.out.print("Enter the index of the element to remove: ");
                int index = scanner.nextInt();
                scanner.nextLine();
                if (index >= 0 && index < linkedList.size()) {
                    linkedList.remove(index);
                    System.out.println("Linked List after removal: " +
linkedList);
                } else {
                    System.out.println("Invalid index!");
                }
            }
        }

        scanner.close();
    }
}
```

Output:

```
Enter number of elements to add:
5
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 4
Enter element 5: 5
Linked List: [1, 2, 3, 4, 5]
First element: 1
Last element: 5
Do you want to change an element? (0/1)
1
Enter the index of the element to change: 3
Enter the new element: 8
Linked List after changing: [1, 2, 3, 8, 5]
Do you want to remove an element? (0/1)
1
Enter the index of the element to remove: 3
Linked List after removal: [1, 2, 3, 5]
```

10. Write a java program to implement a program to create a vector, store the different objects in vector, to display the stored object.

```
import java.util.Scanner;
import java.util.Vector;

public class Vector {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Vector<Object> vector = new Vector<>();

        System.out.println("Enter objects (type 'done' to finish):");
        while (true) {
            String input = scanner.nextLine();
            if (input.equalsIgnoreCase("done")) {
                break;
            }
            vector.add(input);
        }

        System.out.println("\nObjects stored in the Vector:");
        for (int i = 0; i < vector.size(); i++) {
            System.out.println("Element " + (i + 1) + ": " + vector.get(i));
        }

        System.out.println("\nSize of the Vector: " + vector.size());
        System.out.println("Capacity of the Vector: " + vector.capacity());
        System.out.println("Remaining space in the Vector: " + (vector.capacity()
- vector.size()));
        System.out.println("Is the Vector empty? " + vector.isEmpty());

        scanner.close();
    }
}
```

Output:

```
Enter objects (type 'done' to finish):
1
2
3
4
done

Objects stored in the Vector:
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
Size of the Vector: 4
Capacity of the Vector: 10
Remaining space in the Vector: 6
Is the Vector empty? false
```

11. Write an Applet program that automatically displays the text with Font Style, Font type.

Program:

```
import java.awt.*;
import java.applet.*;

public class FontDisplay extends Applet {

  final int PLAIN = Font.PLAIN;
  final int BOLD = Font.BOLD;
  final int ITALIC = Font.ITALIC;

  final String[] fontNames = {"Arial", "Times New Roman", "Courier New"};

  public void init() {
    setBackground(Color.white);
  }

  public void paint(Graphics g) {
    int yPosition = 20;

    for (String fontName : fontNames) {

      for (int style : new int[]{PLAIN, BOLD, ITALIC}) {

        Font font = new Font(fontName, style, 16);
        g.setFont(font);

        String styleName = "Plain";
        if (style == BOLD) {
          styleName = "Bold";
        } else if (style == ITALIC) {
          styleName = "Italic";
        }
```

```
        String text = fontName + " - " + styleName;

        g.drawString(text, 20, yPosition);

        yPosition += 20;
      }
    }
  }
}
```

How to use:

1. Save the code as `FontDisplay.java`.

2. Compile it using a Java compiler: `javac FontDisplay.java`.

3. Create an HTML file (e.g., `index.html`):

index.html

```
<html>
<head>
<title>Font Display Applet</title>
</head>
<body>
<applet code="FontDisplay.class" width="400" height="300">
</applet>
</body>
</html>
```

4. Open `index.html` in a web browser that supports applets (most modern browsers don't support applets by default).

This will display the text with the corresponding font name and style. You can modify the `fontNames` array to include different fonts available on your system.