



**BHARATI VIDYAPEETH'S  
INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT**

(Affiliated to Guru Gobind Singh Indraprastha University,

Approved by AICTE, New Delhi)

# **Object-Oriented Programming and Java (MCA-167) Practical File**

**Submitted To:**

Dr. Ritika Wason

(Associate Professor)

**Submitted By:**

AMAN (T22044032.)

MCA 1<sup>st</sup> Sem, Sec 1/2

## **INDEX**

S. No.	Problem Description	Date of Execution	Sign.
--------	---------------------	-------------------	-------

AP1	<p>Explore the basic java program development scenario in Notepad++ and cmd by creating a class Integer Adder. The adder prints sum of 5 integer numbers without using a single variable where input will be taken through command line arguments.</p> <p>a. Perform the above code using a function and call it in main().</p> <p>a. Make another class and a function in it to perform the above task.</p>	24-11-22	
AP2	<p>Develop a Number Reciprocator java application to computes the sum of the reciprocals in the format:</p> $1/1 + 1/2 + 1/3 + \dots + 1/10$	24-11-22	
Ap3	<p>Demonstrate type conversion in a simple java program by casting and checking output in the following cases:-</p> <p>a. Conversion of int to byte</p> <ul style="list-style-type: none"> <li>• Conversion of double to int</li> <li>• Conversion of double to byte</li> <li>• Conversion of int to char</li> <li>• Conversion of float to short</li> </ul>	24-11-22	
Ap4	<p>Construct a character counter that inputs a piece of text that is analyzed character by character to determine the vowels, spaces and letters used. Fill in the code that computes the number of spaces, vowels, and consonants.</p> <pre> public class StringCharacters {      public static void main(String[] args) {          String text = "To be or not to be, that is the question;"          +"Whether this nobler in the mind to suffer"          +" the slings and arrows of outrageous fortune,"          +" or to take arms against a sea of troubles,"          +" and by opposing end them?";          int spaces = 0, vowels = 0, letters = 0;          //YOUR CODE HERE          System.out.println("The text contained vowels: " + vowels + "\n"         + consonants " + (letters - vowels) + "\n"+ spaces: " + spaces); </pre>	24-11-22	

	<pre> } } </pre>		
Ap5	Construct a number generator to accept three digits (i.e. 0 - 9) and print all its possible combinations. (For example if the three digits are 1, 2, 3 than all possible combinations are: 123, 132, 213, 231, 312, 321.)	26-11-22	
Ap6	A java standalone application makes use of a parameterized method inside a class. Take the following case: Create a class Box and define a method in this class which will return the volume of the box. Initialize two objects for your class and print out the volumes respectively.	26-11-22	
Ap7	A java standalone application reads in a sentence from the user and prints it out with each word reversed, but with the words and punctuation in the original order	28-11-22	
Ap8	<p>Develop an employee pay generator that works on the following rules-</p> <ul style="list-style-type: none"> <li>• An employee gets paid (hours worked) × (base pay), for each hour up to 40 hours.</li> <li>• For every hour over 40, they get overtime = (base pay) × 1.5.</li> </ul> <p>3. The base pay must not be less than the minimum wage (\$8.00 an hour).</p> <p>4. If it is, print an error. If the number of hours is greater than 60, print an error message. //System.err.println();</p>	30-11-22	
Ap9	<p>A Financial Calculator to calculate the SimpleInterest and CompoundInterest by taking command line values for principal, rate and time.</p> <ul style="list-style-type: none"> <li>• Extend the code to calculate 'Final Value' of investment (V) of an investment (principal P) compounded yearly for T years at interest rate R is given by the formula: <math>V = P(1 + R)^T</math></li> <li>• Perform the above code using a function and call it in main().</li> </ul> <p>Make another class and a function in it to perform the above task.</p>	30-11-22	

Bp-1	<p>A group of BVICAM friends decide to run the Airtel Delhi Half Marathon. Their names and times (in minutes) are below:</p> <table><tr><th>Name</th><th>Time (minutes)</th></tr><tr><td>Elena</td><td>341</td></tr><tr><td>Thomas</td><td>273</td></tr><tr><td>Hamilton</td><td>278</td></tr><tr><td>Suzie</td><td>329</td></tr><tr><td>Phil</td><td>445</td></tr><tr><td>Matt</td><td>402</td></tr><tr><td>Alex</td><td>388</td></tr><tr><td>Emma</td><td>275</td></tr><tr><td>John</td><td>243</td></tr><tr><td>James</td><td>334</td></tr><tr><td>Jane</td><td>412</td></tr></table> <p>Find the fastest runner. Print the name and his/her time (in minutes).</p> <p>Optional: Find the second fastest runner. Print the name and his/her time (in minutes).</p> <p>HINT: Define arrays and sort them</p>	Name	Time (minutes)	Elena	341	Thomas	273	Hamilton	278	Suzie	329	Phil	445	Matt	402	Alex	388	Emma	275	John	243	James	334	Jane	412	10-12-22	
Name	Time (minutes)																										
Elena	341																										
Thomas	273																										
Hamilton	278																										
Suzie	329																										
Phil	445																										
Matt	402																										
Alex	388																										
Emma	275																										
John	243																										
James	334																										
Jane	412																										

BP2	<p>Create a class named DuplicateFinder which initializes an array of at least 15 elements. Define appropriate methods to print its elements and calculate duplicate elements if any. It should detail the number of duplicates along with their frequency of occurrence.</p> <p>HINT: count the number of duplicates and print the duplicate element</p>	11-12-2022	
BP3	<p>Define a class named VowelFilter which contains a static method named filterVowel(). This method receives a character array as argument and returns another array which contains only the non-vowel characters of the argument array.</p> <p>HINT: static String remVowel(String str)</p>	12-12-2022	
BP4	<p>Create a class named GradeExam to grade a multiple choice test. There are 20 students and 10 questions in the test. Each row records a student's answers to the questions, as shown in the following array: Grade the students according to their score in the test?</p>	12-12-2022	

BP5	In a GPS navigation system, Given a set of points, the closest-pair problem is to find the two points that are nearest to each other.	13-12-2022	
BP6	Develop a command line driven code to accept the following city name as argument in the command line and sort them in alphabetic order –  City Name = Kolkata, Chennai, Mumbai, Delhi, Bangalore, Ahmedabad.	13-12-2022	
BP7	Create a Waffle Class having two data members flavor and price, being populated by a parameterized constructor. Create another class WaffleMain to compare the values and object context of Waffle class data members by equals() and ==. Extend the implementation to get desired results by overriding  a) equals() b) toString()	15-12-2022	
BP8	Create a POC(Proof of Concept) to demonstrate usage of various functions of String Class like:- a) charAt() b) length() c) contains() d) equals and == e) indexOf() f) split() g) toUpperCase()	15-12-2022	
BP9	Use ragged array to provide the output given below (Take row count from user).	15-12-2022	
CP1	Design a class called DecipherCaesarCode to decipher the Caesar's code.	20-12-2022	
CP2	Create a class called TestPalindromicWord, that prompts user for a word and prints ""xxx" is is not a palindrome".	22-12-2022	
CP3	Create a class to take number of students and their grads and print avg, min, max.	22-12-2022	
CP4	Demonstrate the working of a Static Inner Class through a class Electronics and within it create Static Inner Class Television that has a method cost() which displays cost of television object passed in constructor of Television class. Demonstrate invoking inner class method with outer object when the method cost() is once a :-  a) Instance(Non static) method b) Static method	23-12-2022	

CP5	Simulate a simple banking application. Provide for classes BankAccount. Account will be of two type- Savings and Current. There should be methods to open an account, close an account and perform withdraw, deposit and transfer operations on an account as abstract methods in Account and properly overridden with definition in Account Types. Test classes should instantiate Account Type Classes and provide a menu driven option for operations.	23-12-2022	
CP6	Create a package called animals, let this package contain an interface called Animal that generalizes the eat and travel task of any animal. Implement the given interface in the same package animals and override the methods.	23-12-2022	
CP7	Create a package personpackage. This package contains a class Person with data members to represent firstName and lastName of a person and appropriate methods to read and display the same. Define appropriate class to test the above class outside the above package.	29-12-2022	
CP8	Create an Array of Student class objects. a) Sort them by roll number using Comparable Interface. b) Perform sort based on alphabetic first name numbering using Comparator.	29-01-2023	
CP9	Create an interface Relatable that compares the size of objects. Any class can implement Relatable if there is some way to compare the relative "size" of objects instantiated from the class. For strings, it could be number of characters; for books, it could be number of pages; for students, it could be weight; and so forth. For planar geometric objects, area would be a good choice while volume would work for threedimensional geometric objects. All such classes can implement the isLargerThan() method. Create appropriate implementations.	29-12-2023	
CP10	Create an interface GeoAnalyzer with a constant PI and methods area() and perimeter(). Let Circle, Ellipse and rectangle implement this class. Define a class Geometry that assigns appropriate objects to GeoAnalyzer reference variable.	30-01-2023	
DP1	Extend your Gravity Calculator code (Assignment AA1) to handle exceptions through a try-catch finally block. Handle provision for a divide by zero scenarios caught by NumberFormatException. Explicitly invoke this exception in execution and observe the response.	31-01-2023	
DP2	Further extend your menu-driven Simple Calculator code (Assignment 2) by adding support a Custom Exception code that raises an 'Invalid Numeral' exception each time the user tries to enter any character except a number for calculation. Explicitly invoke this exception in execution and observe the response.	31-01-2023	

DP3	Create a class StudentRegistrationCheck that checks eligibility of a student for registration. If the student age<12 and marks <200. Then student is not eligible for regstrn. Use exception handling.	31-01-2023	
DP4	An ExceptionPOC class is requesting a number between 1 and 10. Run the program again and enter 5.5. Although this number is between 1 and 10, the program will abort. Examine the error message. You should see the word Exception, the method where the exception occurred (main), the class name of the exception (InputMismatchException), as well as the call stack listing the method calls.	01-01-2023	
DP5	Extend CP5 (Bank application)with appropriate exception handling.	01-01-2023	
DP6	Create an Exception class InvalidProductException that can be thrown if a user adds an invalid product.	02-01-2023	
DP7	Compile, run and correct BadThreads.java.	02-01-2023	
DP8	Implement the producer consumer problem using multithreading in java.	03-01-2023	
DP9	Create a Java application that executes concurrent transactions in a bank?	04-01-2023	
DP10	Create a list of numbers and then sort in ascending order as well as in descending order simultaneously.	04-01-2023	
EP1	Model Person with name and age.Manage instances of Person by ensuring that no two instances are duplicated.	08-01-2023	
EP2	Create a subclass of Person (in EP1 above), called ComparablePerson which implements Comparable<Person> interface, and try out the Collections.sort() and Collections.binarySearch() methods on the same.	09-01-2023	
EP3	Model AddressBookEntry that prints name,address and phone of a person. Allow comparison of AddressBookEntry to compare name in a case sensitive manner.	09-01-2023	
EP4	Counts the frequency of each of the words in a file given in the command-line,and save in a map of {word, freq}.	09-01-2023	
FP1	Create a solution to know the address and name of your local machine?	10-01-2023	
FP2	Create a solution to understand the different components of a URL?	10-01-2023	
FP3	Create a connection-less client/server application using UDP protocol that sends system date and time in the format requested by the client.	10-01-2023	

	<p>a. Client: Reads a string representing the required format from the end-user</p> <p>b. Server: returns the system date and time in the requested format or a default format if received format is not understandable</p>		
FP4	<p>Client: Display the required contents. A connection-oriented client/server application using TCP/IP protocol where the client-server has the following responsibilities:</p> <p>a. Server: Creates an Employee class having fields like employeeName, employeeID, and department. Server holds an array of employee objects.</p> <p>b. Client : accepts the employeeID of an employee as an integer from the user.</p> <p>c. Server: Searches for the corresponding employee object, in the array and writes its details to the client stream.</p> <p>d. Client: displays the received object's information</p>	10-01-2023	
FP5	<p>Create a calculator based client/server application where the client sends request to the server in the form of an arithmetic equation of the form "operand1 operator operand2". The server should respond back to answer the equation?</p>	10-01-2023	
GP1	<p>create a class called fruits that has a method mango() that tells if the mango is sweet or sour. Suppose we need a sour mango in taste for only 1 time. Realize this temporary requirement through an anonymous inner class?</p>	10-01-2023	
GP2	<p>Create a class AWTCounter that starts a counter from 0 and increments its value on every button click?</p>	10-01-2023	
GP3	<p>Create the following layout using awt/swing</p> <p>When user clicks the "Display" button the data entered by the user should be displayed in another frame window.</p> <p>When user clicks the "Cancel" button the data fields should get cleared.</p>	10-01-2023	
GP4	<p>Create a class MessageBox that extends Frame. The class should have a constructor that takes a String as a parameter to construct a dialog box that displays the message and OK &amp; CANCEL buttons. The dialog box should get closed when the cancel buttons is clicked. Provide some mechanism in the MessageBox class that can be used by the calling</p>	10-01-2023	



	<p>program to check which button was pressed by the user. The class should have functions to:</p> <p>a) Check which button was pressed by the user.</p> <p>b) Retrieve the string entered by the user, if user pressed OK, null if user pressed CANCEL.</p> <p>Test this class to get a message from a user and display it on a Frame window.</p>		
GP5	Use AWT /Swings to develop a Contact Manager. The GUI application maintains a simple list of contacts storing contact name and number. It provides basic features of CRUD (Create, Update, Retrieve and Delete) as per user choice. The application also supports search feature.	10-01-2023	
GP6	Create a class MyTextEditor to simulate a notepad using Swings?	10-01-2023	
HP1	<p>A String tokenizer application to store the input string contents in a file. Read the file and count vowels, consonants and spaces in each line. Create another file to write the vowel and consonant count besides each line. For eg:- Hi this is java(vowels-5, consonants-7, spaces- 3). I like studying it(vowels-6, consonants-9, spaces-4). Perform this operation using:</p> <p>a) BufferedReader and BufferedWriter</p> <p>b) FileReader and FileWriter</p>	23-01-2023	
HP2	<p>A File Parser a file and store the following text in it-</p> <p>“Dwelling and speedily ignorant any steepest. Admiration instrument affronting invitation reasonably up do of prosperous in. Shy saw declared age debating ecstatic man. Call in so want pure rank am dear were. Remarkably to continuing in surrounded diminution on. In unfeeling existence objection immediate repulsive on he in. Imprudence comparison uncommonly me he difficulty diminution resolution. Likewise proposal differed scarcely dwelling as on raillery. September few dependent extremity own continued and ten prevailed attending. Early to weeks we could.</p> <p>Unpleasant astonished an diminution up partiality. Noisy an their of meant. Death means up civil do an offer wound of. Called square an in afraid direct. Resolution diminution conviction so mr at unpleasing simplicity no. No it as breakfast up conveying earnestly immediate</p>	23-01-2023	

	<p>principle. Him son disposed produced humored overcame she bachelor improved. Studied however out wishing but inhabit fortune windows.”</p> <p>Accept a SearchToken from the user. Open the file and read it using RandomFileAccess and search and display total occurrences of the search string in given text.</p>		
HP3	<p>Define your Student Class to Serialize objects of student class into separate files and byte streams using Serializable interface. Make use of the serialVersionUID field and declare few variables as transient. Deserialize the objects and store them into an array of objects on another JVM instance and perform sorting based on parameter of user’s choice using Comparator.</p> <p>Note: You will have to make separate comparator classes for different data member based sorting.</p>	23-01-2023	
HP4	<p>A Java application that uses JDBC to connect to a database containing table that holds data for students of your class. The application should allow the user to: a) Add a record b) Search for a record c) Modify an existing record Make use of Prepared Statement. Extend the application to call a backend procedure/function. The procedure should take ID field as input parameters and return details of related record</p>	24-01-2023	
HP5	<p>Create an RMI application that exposes a remote object School. It should have two remote methods admit and search. The method admit should add a student’s record in the list of available students and search should return the details of student on basis of roll number entered or raise an exception in case of invalid roll number. Demonstrate the use of these methods in a RMI client application</p>	25-01-2023	

## **Ap:-1**

Explore the basic java program development scenario in Notepad++ and cmd by creating an Integer Adder. The adder prints sum of 5 integer numbers without using single variable where input will be taken through command line arguments.

- Perform the above code using a function and call it in main().
- Make another class and a function in it to perform the above task.

SOLUTION:

```
public class jp1 {
    public static void main(String[] args) {
        num e = new num();
```

```

        e.sum(args);
    }
}

class num{
    public static void sum(String[] args){
        int sum = 0;
        for (int i = 0; i < args.length; i++) {
            sum += Integer.parseInt(args[i]);
        }
        System.out.println(sum);
    }
}

```

### **Output-**

```

C:\WINDOWS\system32\cmd.exe
C:\Users\Aman Yadav\Desktop>javac jp1.java
C:\Users\Aman Yadav\Desktop>java jp1 1 2 3 4 5 6
21
C:\Users\Aman Yadav\Desktop>

```

**AP2.** Develop a Number Reciprocator java application to computes the sum of the reciprocals in the format:

$$1/1 + 1/2 + 1/3 + \dots + 1/10$$

### **Ans-**

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

```
public class jp2 {
    public static void main(String[] args) {
```

```

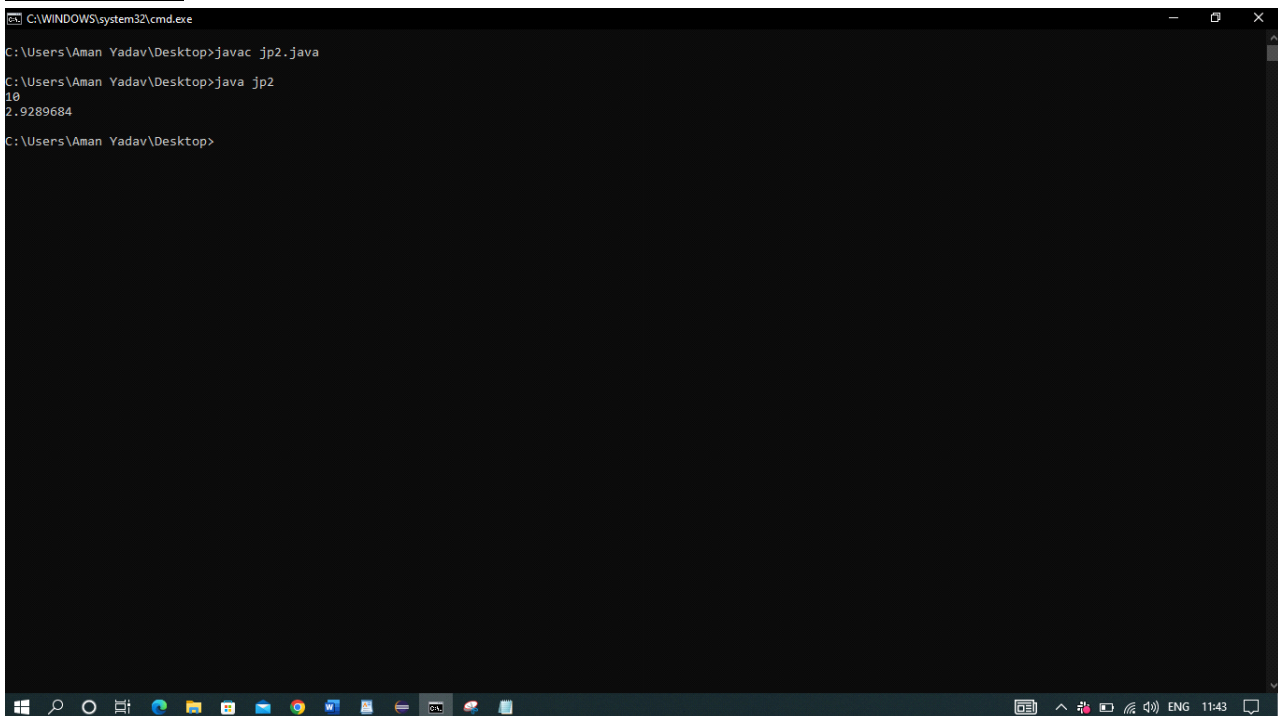
Scanner s = new Scanner(System.in);
int num = s.nextInt();
float sum = 0;
for(int i =1 ; i<=num ; i++){
    float p = (float)1/i;
    sum += p;

}

System.out.println(sum);
}
}

```

### **OUTPUT:-**



```

C:\WINDOWS\system32\cmd.exe
C:\Users\Aman Yadav\Desktop>javac jp2.java
C:\Users\Aman Yadav\Desktop>java jp2
10
2.9289684
C:\Users\Aman Yadav\Desktop>

```

**Ap:-3** Demonstrate type conversion in a simple java program by casting and checking output in the following cases:-

- a. Conversion of int to byte
- b. Conversion of double to int
- c. Conversion of double to byte
- d. Conversion of int to char

Conversion of float to short

**Ans-**

```
import java.util.Scanner;
public class jp3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter Integer value= ");
        int a=sc.nextInt();
        System.out.println("Enter double value= ");
        double c=sc.nextDouble();
        System.out.println("Enter Float value= ");
        float g=sc.nextFloat();


        // int to byte
        byte b=(byte)a;
        System.out.println("Conversion of int to byte = " + b);

        //double to int
        int d=(int)c;
        System.out.println("Conversion of double to int = " + d);

        //double to byte
        byte e=(byte)c;
        System.out.println("Conversion of double to byte = " + e);

        //int to char
        char f=(char)a;
        System.out.println("Conversion of int to char = " + f);

        //float to short
        short h=(short)g;
        System.out.println("Conversion of float to short = " + h);

    }

}
```

**Output-:**

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Aman Yadav\Desktop>java jp3
Enter Integer value=
12
Enter double value=
2.457
Enter Float value=
5.678
Conversion of int to byte = 12
Conversion of double to int = 2
Conversion of double to byte = 2
Conversion of int to char = 0
Conversion of float to short = 5
C:\Users\Aman Yadav\Desktop>
```

**Ap:-4** Construct a character counter that inputs a piece of text that is analyzed character by character to determine the vowels, spaces and letters used. Fill in the code that computes the number of spaces, vowels, and consonants.

**Program:-**

```
public class jp4 {
    public static boolean isvowel(char x){

        if(x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u'){
            return true;
        }
        return false;
    }

    public static void main(String[] args) {

        String text = "To be or not to be, that is the question;"

        +"Whether this nobler in the mind to suffer"

        +" the slings and arrows of outrageous fortune,"

        +" or to take arms against a sea of troubles,"

        +" and by opposing end them?";

        int spaces = 0, vowels = 0, letters = 0;

        text = text.toLowerCase();

        for(char x : text.toCharArray()){

            if(isvowel(x)) vowels++;
```

```

        if(x == ' ') spaces++;
        else if(x >= 'a' && x <= 'z') letters++;

    }
    int c = letters - vowels;
    System.out.println("the text contains\nvowels : " + vowels + "\nconsonants : " + c + "\nspaces : " +
spaces );

}

}

```

## OUTPUT:-

```

C:\WINDOWS\system32\cmd.exe
C:\Users\Aman Yadav\Desktop>javac jp4.java
C:\Users\Aman Yadav\Desktop>java jp4
the text contains
vowels : 60
consonants : 94
spaces : 37
C:\Users\Aman Yadav\Desktop>

```

## Ap:-5

Construct a number generator to accept three digits (i.e. 0 - 9) and print all its possible combinations. (For example if the three digits are 1, 2, 3 than all possible combinations are: 123, 132, 213, 231, 312, 321.)

## Program-

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

```

public class jp5 {
    public static void main(String[] args) {
        int[] arr = new int[3];
        Scanner x = new Scanner(System.in);
        for (int i = 0; i < 3; i++) {
            arr[i] = x.nextInt();
        }
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr.length; j++) {
                for (int j2 = 0; j2 < arr.length; j2++) {
                    if(arr[i] != arr[j] && arr[j] != arr[j2] && arr[i] != arr[j2]){
                        System.out.println(arr[i] + ""+ arr[j] + ""+ arr[j2]);
                    }
                }
            }
        }
    }
}

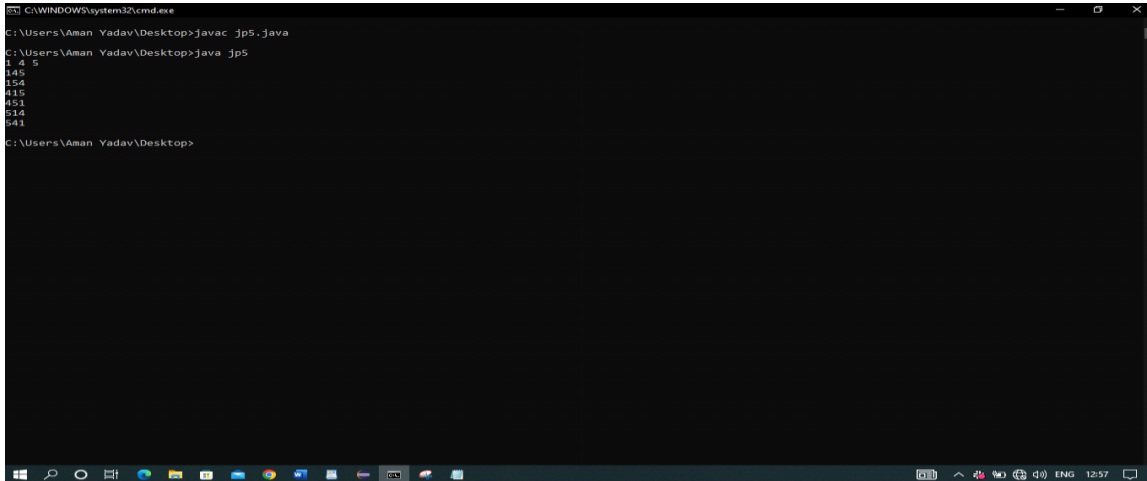
```

```

    }
    }
    }
    }
}

```

### **OUTPUT-:**



```

C:\WINDOWS\system32\cmd.exe
C:\Users\Aman_Yadav\Desktop>javac jp5.java
C:\Users\Aman_Yadav\Desktop>java jp5
1
4
5
14
15
41
45
51
54
541
C:\Users\Aman_Yadav\Desktop>

```

**Ap:-6** A java standalone application makes use of a parameterized method inside a class. Take the following case: Create a class Box and define a method in this class which will return the volume of the box. Initialize two objects for your class and print out the volumes respectively.

### **PROGRAM-:**

```

public class Box {
    public
    float lB,bB,hB;
    public Box(float lB , float bB ,float hB){
        this.lB = lB;
        this.bB= bB;
        this.hB = hB;
    }

    public float getVolume(){
        return lB*bB*hB;
    }
    public static void main(String[] args) {
        Box d = new Box(5f,6f,0.576f);
        Box c = new Box(2f,4f,10);

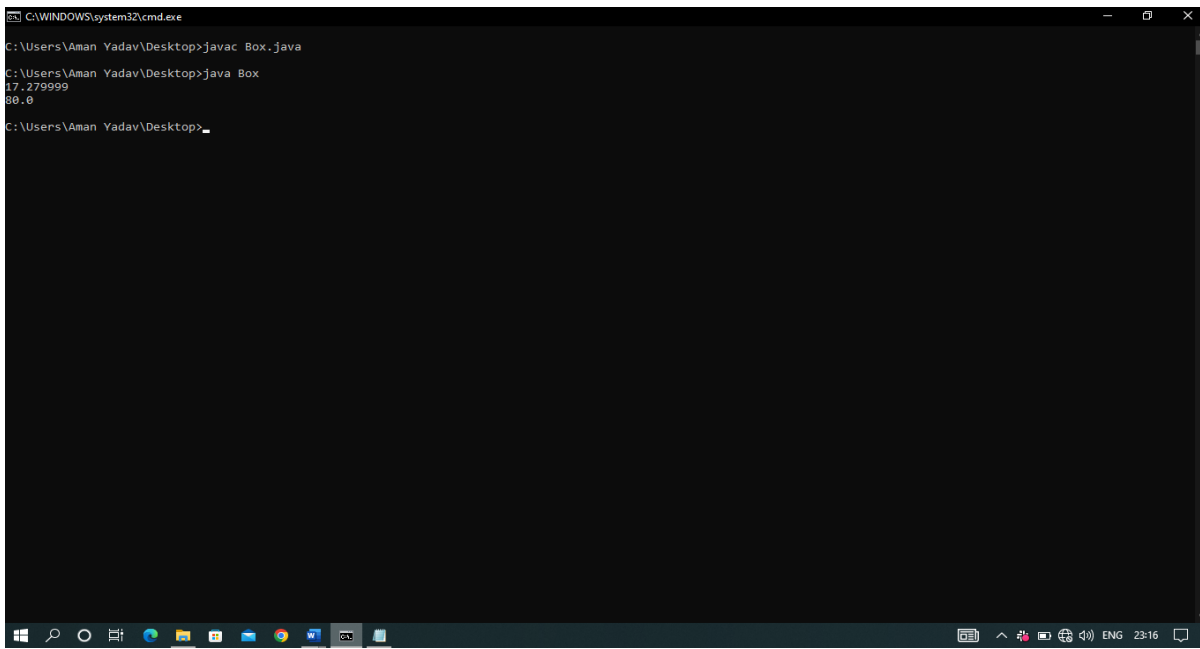
        System.out.println(d.getVolume());
        System.out.println(c.getVolume());

    }
}

```

### **OUTPUT-:**



A screenshot of a Windows command prompt window. The title bar reads 'C:\WINDOWS\system32\cmd.exe'. The command history shows: 'C:\Users\Aman\_Yadav\Desktop>javac Box.java', 'C:\Users\Aman\_Yadav\Desktop>java Box', and the output '17.279999' and '88.8'. The prompt is currently at 'C:\Users\Aman\_Yadav\Desktop>'. The taskbar at the bottom shows various application icons and the system clock indicating 23:16 on ENG.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Aman_Yadav\Desktop>javac Box.java
C:\Users\Aman_Yadav\Desktop>java Box
17.279999
88.8
C:\Users\Aman_Yadav\Desktop>
```

**Ap-7:** A java standalone application reads in a sentence from the user and prints it out with each word reversed, but with the words and punctuation in the original order.

SOLUTION:

```
import java.util.Scanner;

public class Ap7 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String na = sc.nextLine();

        sc.close();

        System.out.println(reverseWord(na));

    }

    public static String reverseWord(String str){

        String words[]=str.split(" ");

        String reverseWord="";

        for(String w:words){

            StringBuilder sb=new StringBuilder(w);

            sb.reverse();
```

```

        reverseWord+=sb.toString()+" ";
    }

    return reverseWord.trim();
}
}

```



```

C:\Users\Aman Yadav\Desktop>javac Ap7.java
C:\Users\Aman Yadav\Desktop>java Ap7
Welcome to java
emocleW ot avaj
C:\Users\Aman Yadav\Desktop>

```

AP 8: Develop an employee pay generator that works on the following rules-

1. An employee gets paid (hours worked) × (base pay), for each hour up to 40 hours.
2. For every hour over 40, they get overtime = (base pay) × 1.5.
3. The base pay must not be less than the minimum wage (\$8.00 an hour).
4. If it is, print an error. If the number of hours is greater than 60, print an error message. `//System.err.println();`

SOLUTION :

```

import java.util.Scanner;

public class Ap8 {
    public static void main(String[] args) {

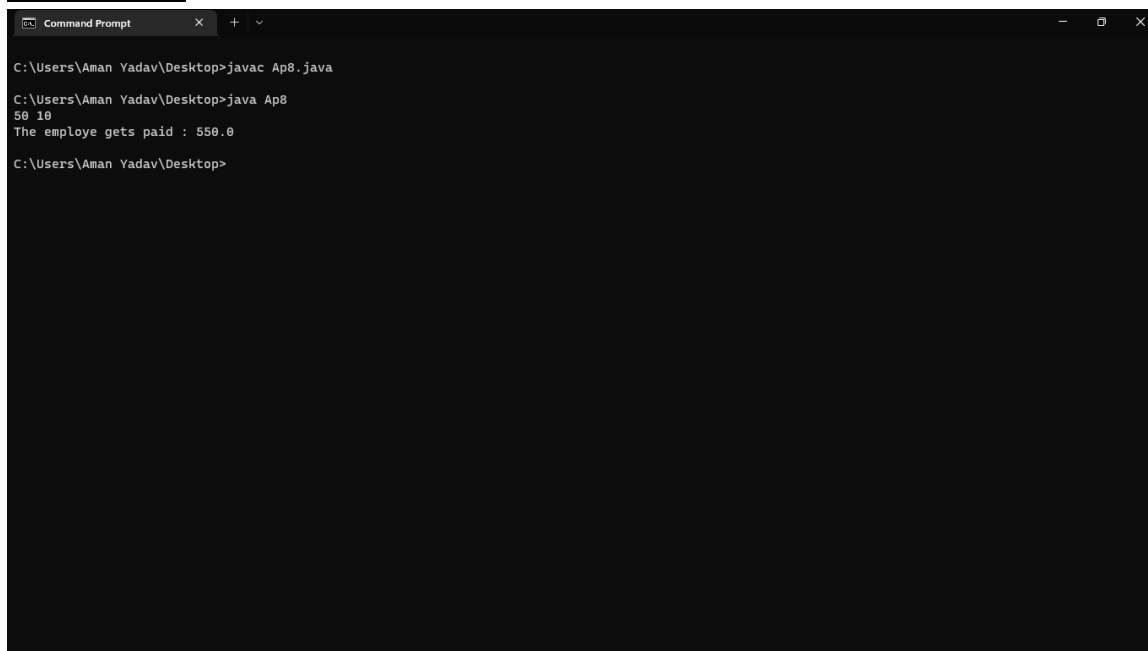
```

```

Scanner sc = new Scanner(System.in);
int hr = sc.nextInt();
float bp = sc.nextFloat();
if(bp < 8) {
    System.out.println("wage should be greater than $8 ");
}
if(hr > 60) {
    System.err.println("no of hours greater than 60");
    System.exit(0);
}
int bhr = 0;
if(hr>40){
    Double bs;
    bhr = hr - 40;
    bs = (40.0 * bp) + (bhr * bp * 1.5);
    System.out.println("The employe gets paid : " + bs );
}
else{
    System.out.println("The employe gets paid : " + hr * bp );
}
}
}

```

## OUTPUT:-



```

C:\Users\Aman Yadav\Desktop>javac Ap8.java
C:\Users\Aman Yadav\Desktop>java Ap8
50 10
The employe gets paid : 550.0
C:\Users\Aman Yadav\Desktop>

```

**Ap:-9** A Financial Calculator to calculate the SimpleInterest and CompoundInterest by taking command line values for principal, rate and time.

1. Extend the code to calculate 'Final Value' of investment (V) of an investment (principal P) compounded yearly for T years at interest rate R is given by the formula:  $V = P (1 + R)^T$
2. Perform the above code using a function and call it in main().  
Make another class and a function in it to perform the above task.

SOLUTION:

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

```

public class ap9{

    public static void main(String[] args){

        double p = Double.parseDouble(args[0]);
        double r = Double.parseDouble(args[1]);
        double t = Double.parseDouble(args[2]);

        double interest = (p * r * t) / 100;
        System.out.println("simple interest : " + interest);

        System.out.println(" to calculate compound interest insert n ");
        Scanner sc = new Scanner(System.in);
        double n = sc.nextDouble();
        sc.close();

        interest = p * (Math.pow((1 + r/100), (t * n))) - p;
        System.out.println("Compound Interest: " + interest);

        double v = p * Math.pow(1 + r , t);
        System.out.println("final value : " + v);

    }
}

```

```

C:\Users\Aman Yadav\Desktop>java Ap9a 2 3 4
simple interest : 0.24
 to calculate compound interest insert n
6
Compound Interest: 2.065588212920808
final value : 512.0
C:\Users\Aman Yadav\Desktop>

```

b)

```

import java.util.Scanner;

public class ap9b {

    public static double si(double p,double r , double t) {
        return (p * r * t) / 100;
    }
    public static double ci(double p,double r , double t,double n) {
        return p * (Math.pow((1 + r/100), (t * n))) - p;
    }
}

```

```

    }
    public static double v(double p,double r , double t) {
        return p * Math.pow(1 + r , t);
    }

    public static void main(String[] args){

        double p = Double.parseDouble(args[0]);
        double r = Double.parseDouble(args[1]);
        double t = Double.parseDouble(args[2]);

        System.out.println("simple interest : " + si(p,r,t));
        System.out.println(" to calculate compound interest insert n ");

        Scanner sc = new Scanner(System.in);
        double n = sc.nextDouble();

        System.out.println("Compound Interest: " + ci(p,r,t,n));
        System.out.println("final value : " + v(p,r,t));
    }
}

```

```

C:\Users\Aman Yadav\Desktop>javac Ap9b.java
C:\Users\Aman Yadav\Desktop>java Ap9b 1.5 5.6 3
simple interest : 0.25199999999999995
 to calculate compound interest insert n
3
Compound Interest: 0.9494384045900062
final value : 431.24399999999997
C:\Users\Aman Yadav\Desktop>

```

## Bp-:1

A group of BVICAM friends decide to run the Airtel Delhi Half Marathon. Their names and times (in minutes) are below:

Name	Time (minutes)
Elena	341
Thomas	273

Hamilton	278
Suzie	329
Phil	445
Matt	402
Alex	388
Emma	275
John	243
James	334
Jane	412

Find the fastest runner. Print the name and his/her time (in minutes).

Optional: Find the second fastest runner. Print the name and his/her time (in minutes).

**HINT: Define arrays and sort them**

### Program-:

```
class Bp1{
    public static int min(int[] arr,int n){
        int max = 0;
        for(int i =0;i<n;i++){
            if(arr[max] > arr[i]){
                max = i;
            }
        }
        return max;
    }

    public static int smin(int[] arr,int n,int max){
        int smax = 0;
        for(int i =0;i<n;i++){
            if(i != max && arr[smax] > arr[i]){
                smax = i;
            }
        }
    }
}
```

```

    }

    }

    return smax;

}

public static void main(String[] args){

    String[] na =
{"Elina","Thomas","Hamilton","Suzie","Phil","Matt","Alex","Emma","John",
"James","Jane"};

    int[] arr = {341,273,278,329,445,402,388,275,243,334,412};

    System.out.print("the fastest runner is ");

    int min = min(arr,arr.length);

    System.out.print(na[min]);

    System.out.println(" " + arr[min]);


    System.out.print("the second fastest runnner is ");

    int smin = smin(arr,arr.length,min);

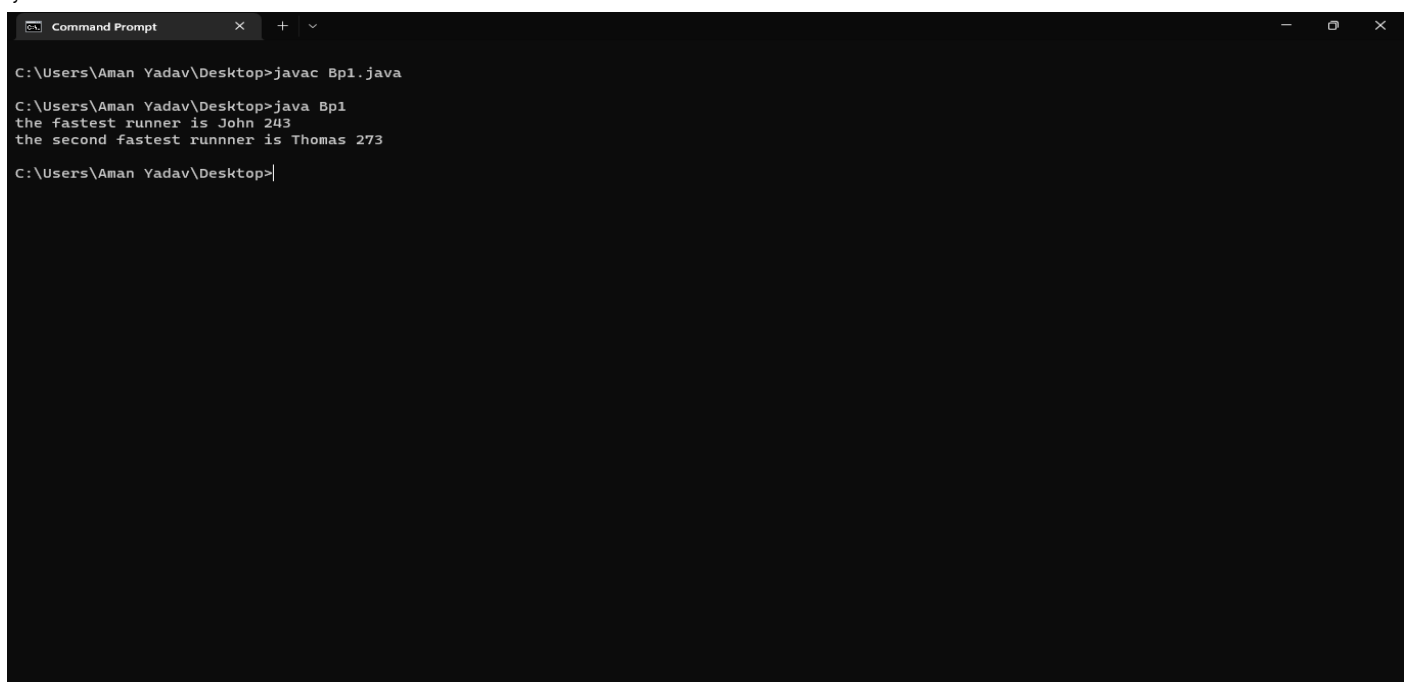
    System.out.print(na[smin]);

    System.out.println(" " + arr[smin]);

}

}

```



```

C:\Users\Aman Yadav\Desktop>javac Bp1.java
C:\Users\Aman Yadav\Desktop>java Bp1
the fastest runner is John 243
the second fastest runnner is Thomas 273
C:\Users\Aman Yadav\Desktop>

```

## BP:-2

Create a class named DuplicateFinder which initializes an array of at least 15 elements. Define appropriate methods to print its elements and calculate duplicate elements if any. It should detail the number of duplicates along with their frequency of occurrence.

PROGRAM:-

```
import java.util.*;

public class DuplicateFinder
{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the length of the array");
        int n=sc.nextInt();
        int a[]=new int[n];
        System.out.println("Enter the array elements ");
        for (int i=0;i<n;i++) {
            a[i]=sc.nextInt();
        }
        HashMap<Integer,Integer> map=new HashMap<>();
        for(int i=0;i<n;i++) {
            if(map.containsKey(a[i]))
            {
                int c=map.get(a[i]);
                map.replace(a[i],c+1);
            }
            else
                map.put(a[i], 1);
        }
        System.out.println("Elements Frequency");
```



```

        for(Map.Entry<Integer,Integer> i : map.entrySet())
        {
            if(i.getValue()>1)
                System.out.println(" " + i.getKey() + "    " + i.getValue());
            else
                continue;
        }
    }
}

```

## OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>javac DuplicateFinder.java

C:\Users\Aman Yadav\Desktop>java DuplicateFinder
Enter the length of the array
6
Enter the array elements
88
55
12
68
12
55
Elements Frequency
55      2
12      2

```

## Bp:-3

Define a class named VowelFilter which contains a static method named filterVowel(). This method receives a character array as argument and returns another array which contains only the non-vowel characters of the argument array.

### Program-:

```

import java.util.Scanner;

public class VowelFilter {

    static char[] filterVowel(char[] text){

        char[] ans = new char[text.length];

        int j = 0 ;

        for(int i = 0; i < text.length; i++){

```

```

        if(text[i] == 'a' || text[i]== 'e' || text[i]== 'i' || text[i]== 'o' ||
            text[i]== 'u' || text[i] == 'A' || text[i]== 'E' || text[i]== 'I' || text[i]== 'O'
            || text[i]== 'U'){
continue;
    }
    else{
        ans[j] = text[i];
        j++;
    }
}

return ans;
}

public static void main(String[] args) {
    char[] arr = new char[10];

    Scanner S = new Scanner(System.in);

    System.out.println("Enter array elements");

    for(int i = 0; i < 10; i++){
        arr[i] = S.nextLine().charAt(0);
    }

    S.close();

    System.out.println(filterVowel(arr));
}

```

## OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java VowelFilter
Enter array elements
m
a
s
g
i
o
t
r
e
w
msgtrw

```

## Bp-:4

**Create a class named GradeExam to grade a multiple choice test. There are 20 students and 10 questions in the test. Each row records a student's answers to the questions, as shown in the following array: Grade the students according to their score in the test?**

**Program-:**

```
public class GradeExam
{
    public static void main(String args[]) {
        char[][] answers = {
            {'A', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
            {'D', 'B', 'A', 'B', 'C', 'A', 'E', 'E', 'A', 'D'},
            {'E', 'D', 'D', 'A', 'C', 'B', 'E', 'E', 'A', 'D'},
            {'C', 'B', 'A', 'E', 'D', 'C', 'E', 'E', 'A', 'D'},
            {'A', 'B', 'D', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
            {'B', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
            {'B', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
            {'E', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'}};

        // Key to the questions
        char[] keys = {'D', 'B', 'D', 'C', 'C', 'D', 'A', 'E', 'A', 'D'};

        // Grade all answers
        for (int i = 0; i < answers.length; i++) {

            // Grade one student
            int correctCount = 0;

            for (int j = 0; j < answers[i].length; j++) {
                if (answers[i][j] == keys[j])
                    correctCount++;
            }

            System.out.println("Student " + i + "'s correct count is " +
```

```
        correctCount);  
    }  
}  
}
```

## OUTPUT-:

```
C:\Users\Aman Yadav\Desktop>javac GradeExam.java  
  
C:\Users\Aman Yadav\Desktop>java GradeExam  
Student 0's correct count is 7  
Student 1's correct count is 6  
Student 2's correct count is 5  
Student 3's correct count is 4  
Student 4's correct count is 8  
Student 5's correct count is 7  
Student 6's correct count is 7  
Student 7's correct count is 7
```

## Bp:-5

**In a GPS navigation system, Given a set of points, the closest-pair problem is to find the two points that are nearest to each other**

### Program-:

```
import java.util.Scanner;  
  
public class GPS {  
  
    public static void main(String[] args) {  
  
        int i,j;  
  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter the number of points :");  
  
        int n = sc.nextInt();  
  
        double[][] p = new double[n][2];  
  
        System.out.printf("\nEnter %d points",n);  
  
        for( i=0;i<p.length;i++){  
  
            p[i][0] = sc.nextDouble();  
  
            p[i][1] = sc.nextDouble();  

```

```

    }

    int p1 = 0, p2 = 1;

    double shortDis = distance(p[p1][0],p[p1][1],p[p2][0],p[p2][1]);

    for( i=0;i<p.length;i++){

        for(j=i+1;j<p.length;j++){

            double distance = distance(p[i][0],p[i][1],p[j][0],p[j][1]);

            if(shortDis>distance){

                p1 = i;

                p2 = j;

                shortDis = distance;

            }

        }

    }

    System.out.println("Closest 2 points are"+"("+p[p1][0]+","+p[p1][1]+") and ("
    +p[p2][0]+","+p[p2][1]+")");

}

public static double distance(double x1,double y1,double x2,double y2){

    return Math.sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));

}

}

```

## OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java GPS
Enter the number of points :
4

Enter 4 points
4 5
6 8
7 6
8 4
Closest 2 points are(6.0,8.0) and (7.0,6.0)

```

## Bp-6

Develop a command line driven code to accept the following city name as argument in the command line and sort them in alphabetic order – City Name = Kolkata, Chennai, Mumbai, Delhi, Bangalore, Ahmedabad.

## Program:-

```
import java.util.Scanner;

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

    int count; String temp;

    Scanner scan = new Scanner(System.in);

    System.out.print("Enter number of strings you would like to enter:");

    count = scan.nextInt();

    String str[] = new String[count];

    Scanner scan2 = new Scanner(System.in);

    System.out.println("Enter the Strings one by one:");

    for(int i = 0; i < count; i++) { str[i] = scan2.nextLine();

    }

    scan.close();

    scan2.close();

    for (int i = 0; i < count; i++) {

        for (int j = i + 1; j < count; j++) {

            if (str[i].compareTo(str[j])>0) {

                temp = str[i];

                str[i] = str[j];

                str[j] = temp;

            }

        }

    }

    System.out.print("Strings in Sorted Order:");

    for (int i = 0; i <= count - 1; i++) {

        System.out.print(str[i] + ", ");

    }

}
```

OUTPUT:-

```
Enter number of strings you would like to enter:6
Enter the Strings one by one:
Kolkata
Chennai
Mumbai
Delhi
Banglore
Ahemdabad
Strings in Sorted Order:Ahemdabad, Banglore, Chennai, Delhi, Kolkata, Mumbai,
```

### **Bp:-7**

Create a Waffle Class having two data members flavor and price, being populated by a parameterized constructor. Create another class WaffleMain to compare the values and object context of Waffle class data members by equals() and ==. Extend the implementation to get desired results by overriding

a) equals()

b) toString()

**Program:-**

```
public class WaffleMain{

    public static void main(String[] args) {

        Waffle waffle1 = new Waffle("Choco",200D);

        Waffle waffle2 = new Waffle("Choco",200D);

        System.out.println(waffle1.equals(waffle2));

        System.out.println(waffle1==waffle2);

    }

}

class Waffle{

    private String flavour;

    private Double price;

    public Waffle(String flavour, Double price) {

        this.flavour = flavour;

        this.price = price;

    }

    public String getFlavour()

    {

        return flavour;

    }

}
```

```
    public void setFlavour(String flavour) {  
  
this.flavour = flavour;  
  
    }  
  
    public Double getPrice() {  
  
return price;  
  
    }  
  
    public void setPrice(Double price) {  
  
this.price = price;  
  
    }  
  
    public boolean equals(Waffle waffle){  
  
        if (this.flavour.equals(waffle.flavour) && this.price.equals(waffle.price))  
  
        {  
  
            return true;  
  
        }  
  
else  
  
        {  
  
            return false;  
  
        }  
  
    }  
  
}
```

**OUTPUT:-**

```
C:\Users\Aman Yadav\Desktop>java WaffleMain  
true  
false
```

## **Bp:-8**

Create a POC(Proof of Concept) to demonstrate usage of vari functions of String Class like:-

- a) charAt()
- b) length()
- c) contains()
- d) equals and ==
- e) indexOf()



f) split()

g) toUpperCase()

### **Program-:**

```
public class bp8 {  
  
    public static void main(String[] args) {  
  
        String test1 = "abcdef";  
  
        String test2 = "abc def";  
  
        String test4 = "abc";  
  
        String test3 = new String("abcdef");  
  
  
        //length()  
  
        System.out.println(test1.length());  
  
  
        //contains()  
  
        System.out.println(test1.contains(test4));  
  
  
        //equals()  
  
        System.out.println(test1.equals(test3));  
  
  
        //==  
  
        System.out.println(test1 == test3);  
  
  
        //indexOf()  
  
        System.out.println(test1.indexOf('d'));  
  
  
        //split()  
  
        String[] ch = test2.split(" ");  
  
        for (String x : ch) {  
  
            System.out.println(x);  
  
        };  
    }  
}
```

```

//toUpperCase()

System.out.println(test4.toUpperCase());

//charAt

System.out.println(test3.charAt(1));

}

}

```

**OUTPUT:-**

```

C:\Users\Aman Yadav\Desktop>java StringComparison
6
true
true
false
3
abc
def
ABC
b

```

**BP-9:**

Use ragged array to provide the output given below (Take row count from user).

**PROGRAM:-**

```

public class Pattern{

    public static void main(String[] args)

    {

        int i,j;

        for (i=1;i<=7;i++)

        {

            for(int k=1;k<=6;k++)

            {

                System.out.print(" ");

            }

            for(j=1;j<=i;j++)

```

```

{
    System.out.print(j);
}

System.out.println();
}

for(i=1;i<=7;i++)
{
    int k=1;

    for(j=1;j<=7;j++)
    {
        if(i+j > 7)
        {
            System.out.print(k);

            k++;
        }
    }
    else
        System.out.print(" ");
}
System.out.println();
}
}
}

```

**OUTPUT:-**

```

C:\Users\Aman Yadav\Desktop\BP9>java BP9
1
12
123
1234
12345
123456
1234567
1
12
123
1234
12345
123456
1234567

```

## CP-1

**Design a class called DecipherCaesarCode to decipher the Caesar's code. The program shall prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase. Design the solution with appropriate methods.**

### Program-:

```
public class CP1m {

    public static void main(String[] args) {

        String Encoded =

            DecipherCaesarCode.Encode("HELLOWORLD", 3);

        System.out.println("Encoded String: " +

            Encoded);

        System.out.println("Decoded String: " +

            DecipherCaesarCode.Decode(Encoded, 3));

    }

}

class DecipherCaesarCode {

    public static String Encode(String str, int n)

        throws IllegalArgumentException {

        String encoded = "";

        for(int i = 0; i < str.length(); i++) {

            if (str.charAt(i) >= 'A' && str.charAt(i) <= 'Z') {

                int c = (str.charAt(i) - 'A' + n) % 26;

                encoded += (char) (c + 'A');

            }

            else throw

                new IllegalArgumentException("Char range A-Z got: " +

                    str.charAt(i));

        }

        return encoded;

    }

}
```

```

    }

    public static String Decode(String str, int n)

        throws IllegalArgumentException {

        String decoded = "";

        for(int i = 0; i < str.length(); i++) {

            if (str.charAt(i) >= 'A' && str.charAt(i) <= 'Z') {

                int c = (str.charAt(i) - 'A' - n) % 26;

                decoded += (char) (c + 'A');

            }

            else throw

                new IllegalArgumentException("Char range A-Z got: " +

                    str.charAt(i));

        }

        return decoded;

    }

}

```

#### OUTPUT:-

```

C:\Users\Aman Yadav\Desktop>java CP1m
Encoded String: KHOORZRUOG
Decoded String: HELLOWORLD

```

**CP2:** A word that reads the same backward as forward is called a palindrome, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive). Create a class called TestPalindromicWord, that prompts user for a word and prints ""xxx" is|is not a palindrome".

#### SOLUTION:

```

import java.util.Scanner;

public class CP2 {

    public static void main(String[] args) {

        TestPalindromicWord.checkPlaindrome();
    }
}

```

```

    }
}

class TestPalindromicWord {
    public static void checkPlaindrome() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the word: ");
        String word = sc.next();
        sc.close();

        boolean isPlaindrome = true;
        for(int i = 0; i < word.length() / 2; i++) {
            if (word.charAt(i) != word.charAt(word.length() - i - 1)) {
                isPlaindrome = false;
                break;
            }
        }

        System.out.println(
            "\"" + word + "\" is " +
            (isPlaindrome ? "" : "not ") +
            "a palindrome"
        );
    }
}

```

#### OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java CP2
Enter the word: malayalam
"malayalam" is a palindrome

```

**CP3:** A java based program which prompts user for the number of students in a class (a non-negative integer), and saves it in an int variable called numStudents. It then prompts user for the grade of each of the students (integer between 0 to 100) and saves them in an int array called grades. The program shall then compute and print the average (in double rounded to 2 decimal places) and minimum/maximum (in int).

#### SOLUTION:

```
import java.text.DecimalFormat;

import java.util.Scanner;

public class CP3 {

    public static void main(String[] args) throws Exception{

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of students: ");

        int numStudents = sc.nextInt();

        if(numStudents <= 0) {

            sc.close();

            throw new IllegalArgumentException(

                "Number of students must be greater than zero"

            );

        }

        int[] numGrade = new int[numStudents];

        System.out.print("Enter the grades (0, 100): ");

        int i = 0;

        sc.nextLine();

        String grades = sc.nextLine();

        Scanner parser = new Scanner(grades);

        while(parser.hasNextInt() && i < numStudents) {

            numGrade[i] = parser.nextInt();

            if(numGrade[i] < 0 || numGrade[i] > 100) {

                sc.close();

                parser.close();

                throw new IllegalArgumentException(

                    "Grade must be between 0 and 100"

                );

            }

            i++;

        }

        parser.close();

        sc.close();

    }

}
```

```

int max = numGrade[0];
int min = numGrade[0];
double sum = numGrade[0];
for(i = 1; i < numStudents; i++) {
    max = Math.max(max, numGrade[i]);
    min = Math.min(min, numGrade[i]);
    sum += numGrade[i];
}
DecimalFormat df = new DecimalFormat("##.00");
System.out.println(
    "Average: " + df.format(sum / numStudents) + "\n" +
    "Maximum: " + max + "\n" +
    "Minimum: " + min
);
}
}

```

#### OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java CP3
Enter the number of students: 90
Enter the grades (0, 100): 90
Average: 1.00
Maximum: 90
Minimum: 0

```

**CP4:** Demonstrate the working of a Static Inner Class through a class Electronics and within it create Static Inner Class Television that has a method cost() which displays cost of television object passed in constructor of Television class. Demonstrate invoking inner class method with outer object when the method cost() is once a :-

- a) Instance(Non static) method
- b) Static method

#### SOLUTION:

```

public class CP4 {
    public static void main(String[] args) {
        Electronics.Television t = new Electronics.Television(5);
    }
}

```



```

        t.cost();
        Electronics.Television.cost(t);
    }
}

```

```

class Electronics {
    static class Television {
        private int cost = 0;
        Television(int cost) {
            this.cost = cost;
        }

        void cost() {
            System.out.println("cost(instance): " + cost);
        }

        static void cost(Television t) {
            System.out.println("cost(static): " + t.cost);
        }
    }
}

```

```

PS C:\Users\Rahul gupta\Desktop\CP\CP4> java CP4.java
cost(instance): 5
cost(static): 5

```

**CP4:** Demonstrate the working of a Static Inner Class through a class Electronics and within it create Static Inner Class Television that has a method cost() which displays cost of television object passed in constructor of Television class. Demonstrate invoking inner class method with outer object when the method cost() is once a :-

- a) Instance(Non static) method
- b) Static method

## **SOLUTION:**

```
public class CP4 {  
    public static void main(String[] args) {  
        Electronics.Television t = new Electronics.Television(5);  
        t.cost();  
        Electronics.Television.cost(t);  
    }  
}
```

```
class Electronics {  
    static class Television {  
        private int cost = 0;  
        Television(int cost) {  
            this.cost = cost;  
        }  
  
        void cost() {  
            System.out.println("cost(instance): " + cost);  
        }  
  
        static void cost(Television t) {  
            System.out.println("cost(static): " + t.cost);  
        }  
    }  
}
```

## **OUTPUT:-**

```
C:\Users\Aman Yadav\Desktop>javac CP4.java  
  
C:\Users\Aman Yadav\Desktop>java CP4  
cost(instance): 5  
cost(static): 5
```

**CP5:** Simulate a simple banking application. Provide for classes BankAccount. Account will be of two type- Savings and Current. There should be methods to open an account, close an account and perform withdraw, deposit and transfer operations on an account as abstract methods in Account and properly overridden with definition in Account Types. Test classes should instantiate Account Type Classes and provide a menu driven option for operations.

**SOLUTION:**

```
import java.util.HashMap;
import java.util.Scanner;

public class CP5 {
    static HashMap<Integer, BankAccount> accounts =
        new HashMap<Integer, BankAccount>();

    public static void addAccount(int accountNumber, int type) {
        switch (type) {
            case 1:
                accounts.put(
                    accountNumber,
                    new SavingsAccount(accountNumber, 0)
                );
                break;
            case 2:
                accounts.put(
                    accountNumber,
                    new CurrentAccount(accountNumber, 0)
                );
                break;
        }
    }

    public static void main(String[] args) {
```

```
int an = 1001;

String menu = "\n" +
    "1. Create a new bank account \n" +
    "2. Add money to the account \n" +
    "3. Withdraw money from account \n" +
    "4. Transfer money to bank account \n" +
    "5. Get the balance \n" +
    "6. Exit \n" +
    "Enter your choice: ";

String menu2 =
    "Type of bank account: \n" +
    "  1. Savings account \n" +
    "  2. Current account \n" +
    "Enter your choice: ";

Scanner sc = new Scanner(System.in);

boolean isRunning = true;

while (isRunning) {
    System.out.print(menu);
    int choice = sc.nextInt();
    switch (choice) {
        case 1: {
            System.out.print(menu2);
            int type = sc.nextInt();
            addAccount(an, type);
            System.out.println("Your account no is " + an++);
            break;
        }
        case 2: {
            System.out.print("Enter the account_no: ");
            int account_number = sc.nextInt();
```

```
        System.out.print("Enter the amount: ");
        float amount = sc.nextFloat();
        accounts.get(account_number).deposit(amount);
        break;
    }
    case 3: {
        System.out.print("Enter the account_no: ");
        int account_number = sc.nextInt();
        System.out.print("Enter the amount: ");
        float amount = sc.nextFloat();
        accounts.get(account_number).withdraw(amount);
        break;
    }
    case 4: {
        System.out.print("Enter the account_no: ");
        int account_number = sc.nextInt();
        System.out.print("Enter 2nd account_no: ");
        int r_account_number = sc.nextInt();
        System.out.print("Enter the amount: ");
        float amount = sc.nextFloat();
        accounts.get(account_number)
            .transfer(
                accounts.get(r_account_number),
                amount
            );
        break;
    }
    case 5: {
        System.out.print("Enter the account_no: ");
        int account_number = sc.nextInt();
```

```
        System.out.println(
            "balance: " +
            accounts.get(account_number)
        );
        break;
    }
    case 6: {
        sc.close();
        isRunning = false;
        break;
    }
}
}
```

```
class BankAccount {
    int account_id;
    float balance;

    public BankAccount(int account_id, float balance) {
        this.account_id = account_id;
        this.balance = balance;
    }

    public float getBalance() {
        return balance;
    }

    public float getAccountId() {
```

```
    return account_id;
}
```

```
public void deposit(float amount) {
    System.out.println("Depositing: " + amount +
        " In account " + account_id);
    balance += amount;
}
```

```
public boolean withdraw(float amount) {
    if (amount > balance) {
        System.out.println("Low balance: "
            + balance +
            " Can't withdraw");
        return false;
    }
    System.out.println("withdrawing: " + amount +
        " From: " + account_id);
    balance -= amount;
    return true;
}
```

```
public boolean transfer(BankAccount account, float amount) {
    if (amount > balance) {
        System.out.println("Low balance: " + balance + " Can't transfer");
        return false;
    }
    System.out.println("transferring: " + amount +
        "\nFrom: " + account_id +
        "\nTo: " + account.account_id);
}
```

```
        account.balance += amount;
        balance -= amount;
        return true;
    }
}
```

```
class SavingsAccount extends BankAccount {
    static int max_transactions = 10, max_withdraws = 10;
    int transactions = 0;
    int withdraws = 0;

    SavingsAccount(int account_id, float balance) {
        super(account_id, balance);
    }
}
```

@Override

```
public boolean withdraw(float amount) {
    if (withdraws == max_withdraws) {
        System.out.println("number of withdraws exceeded");
        return false;
    }
    withdraws++;
    return super.withdraw(amount);
}
```

@Override

```
public boolean transfer(BankAccount account, float amount) {
    if (transactions == max_transactions) {
        System.out.println("number of transfers exceeded");
        return false;
    }
}
```



```

    }
    transactions++;
    return super.transfer(account, amount);
}
}

class CurrentAccount extends BankAccount {
    CurrentAccount(int account_id, float balance) {
        super(account_id, balance);
    }
}

```

**OUTPUT-:**

```

1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 1
Type of bank account:
    1. Savings account
    2. Current account
Enter your choice: 1
Your account no is 1001

1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 1
Type of bank account:
    1. Savings account
    2. Current account
Enter your choice: 1
Your account no is 1002

```

```
1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 2
Enter the account_no: 1001
Enter the amount: 10000
Depositing: 10000.0 In account 1001
```

```
1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 3
Enter the account_no: 1001
Enter the amount: 2000
withdrawing: 2000.0 From: 1001
```

```
1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 4
Enter the account_no: 1001
Enter 2nd account_no: 1002
Enter the amount: 3000
transferring: 3000.0
From: 1001
To: 1002
```

```
1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 5
Enter the account_no: 1001
balance: SavingsAccount@38c6f217
```

```
1. Create a new bank account
2. Add money to the account
3. Withdraw money from account
4. Transfer money to bank account
5. Get the balance
6. Exit
Enter your choice: 5
```

**CP6:** Create a package called animals, let this package contain an interface called Animal that generalizes the eat and travel task of any animal. Implement the given interface in the same package animals and override the methods appropriately.

**SOLUTION:**

```
import animals.animals;
```

```
public class CP6 {
```

```

public static void main(String[] args) {
    animals.Dog d = new animals.Dog();
    animals.Cat c = new animals.Cat();

    d.eat();
    d.travel();

    c.eat();
    c.travel();
}
}

```

**OUTPUT-:**

```

Eating dog food
Chasing the cat
Eating cat food
Running away from the dog

```

**CP7:** Create a package personpackage. This package contains a class Person with data members to represent firstName and lastName of a person and appropriate methods to read and display the same. Define appropriate class to test the above class outside the above package.

**SOLUTION:**

```

import personpackage.Person;

public class CP7 {
    public static void main(String[] args) {
        Person person = new Person("John", "Doe");
        person.display();
    }
}

```

**OUTPUT-:**

```

First name: John
Last name: Doe

```

**CP8:** Create an Array of Student class objects.

- a) Sort them by roll number using Comparable Interface.
- b) Perform sort based on alphabetic first name numbering using Comparator

**SOLUTION:**

```
import java.util.Arrays;
import java.util.Comparator;

public class CP8 {

    public static void main(String[] args) {

        Student[] students = {

            new Student(1, "John", "Doe"),
            new Student(3, "Bob", "Smith"),
            new Student(2, "Jane", "Doe"),
            new Student(4, "Alice", "Smith")

        };

        // sort students by roll number using Comparable interface
        Arrays.sort(students);

        System.out.println("Student " + Arrays.toString(students));

        // sort students by first name using Comparator interface
        Arrays.sort(students, Student.FirstNameComparator);

        System.out.println("Student " + Arrays.toString(students));

    }

}
```

```
class Student implements Comparable<Student> {

    private int rollNumber;
```

```
private String firstName;
```

```
private String lastName;
```

```
public Student(int rollNumber, String firstName, String lastName) {  
    this.rollNumber = rollNumber;  
    this.firstName = firstName;  
    this.lastName = lastName;  
}
```

```
public int getRollNumber() {  
    return rollNumber;  
}
```

```
public String getFirstName() {  
    return firstName;  
}
```

```
public String getLastName() {  
    return lastName;  
}
```

```
@Override
```

```
public int compareTo(Student other) {  
    return Integer.compare(rollNumber, other.rollNumber);  
}
```

```
public static Comparator<Student> FirstNameComparator =  
    new Comparator<Student>() {  
        @Override  
        public int compare(Student s1, Student s2) {
```

```
        return s1.firstName.compareTo(s2.firstName);
    }
};

public String toString() {
    return String.format(
        "Student[\n\trollNumber=%d,\n\tfirstName=%s,\n\tlastName=%s\n]",
        rollNumber, firstName, lastName
    );
}
}
```

OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java CP8
Student [Student[
    rollNumber=1,
    firstName=John,
    lastName=Doe
], Student[
    rollNumber=2,
    firstName=Jane,
    lastName=Doe
], Student[
    rollNumber=3,
    firstName=Bob,
    lastName=Smith
], Student[
    rollNumber=4,
    firstName=Alice,
    lastName=Smith
]]
Student [Student[
    rollNumber=4,
    firstName=Alice,
    lastName=Smith
], Student[
    rollNumber=3,
    firstName=Bob,
    lastName=Smith
], Student[
    rollNumber=2,
    firstName=Jane,
    lastName=Doe
], Student[
    rollNumber=1,
    firstName=John,
    lastName=Doe
]]

```

**CP9:** Create an interface Relatable that compares the size of objects. Any class can implement Relatable if there is some way to compare the relative "size" of objects instantiated from the class. For strings, it could be number of characters; for books, it could be number of pages; for students, it could be weight; and so forth. For planar geometric objects, area would be a good choice while volume would work for threedimensional geometric objects. All such classes can implement the `isLargerThan()` method. Create appropriate implementations.

**SOLUTION:**

```

public class CP9 {
    public static void main(String[] args) {
        StringObject str1 = new StringObject("hello");

```

```
StringObject str2 = new StringObject("world");
```

```
Book book1 = new Book("The Great Gatsby", 180);
```

```
Book book2 = new Book("War and Peace", 1225);
```

```
System.out.println(str1.isLargerThan(str2)); // false
```

```
System.out.println(book1.isLargerThan(book2)); // false
```

```
System.out.println(book2.isLargerThan(book1)); // true
```

```
}
```

```
}
```

```
interface Relatable {
```

```
    boolean isLargerThan(Relatable other);
```

```
}
```

```
class StringObject implements Relatable {
```

```
    private String str;
```

```
    public StringObject(String str) {
```

```
        this.str = str;
```

```
}
```

```
    public String getString() {
```

```
        return str;
```

```
}
```

```
@Override
```

```
    public boolean isLargerThan(Relatable other) {
```

```
        if (other instanceof StringObject) {
```

```
            StringObject otherString = (StringObject) other;
```

```
            return str.length() > otherString.getString().length();
```



```
    }  
    return false;  
}  
}
```

```
class Book implements Relatable {  
    private String title;  
    private int numPages;  
  
    public Book(String title, int numPages) {  
        this.title = title;  
        this.numPages = numPages;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
  
    public int getNumPages() {  
        return numPages;  
    }  
  
    @Override  
    public boolean isLargerThan(Relatable other) {  
        if (other instanceof Book) {  
            Book otherBook = (Book) other;  
            return numPages > otherBook.getNumPages();  
        }  
        return false;  
    }  
}
```

```
}
```

OUTPUT-:

```
C:\Users\Aman Yadav\Desktop>java CP9  
false  
false  
true
```

**CP10:** Create an interface GeoAnalyzer with a constant PI and methods area() and perimeter(). Let Circle, Ellipse and rectangle implement this class. Define a class Geometry that assigns appropriate objects to GeoAnalyzer reference variable.

**SOLUTION:**

```
public class CP10 {  
    public static void main(String[] args) {  
        Geometry.run();  
    }  
}  
  
interface GeoAnalyzer {  
    double PI = 3.14159265358979323846;  
    double area();  
    double perimeter();  
}  
  
class Circle implements GeoAnalyzer {  
    private double radius;  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
    @Override  
    public double area() {  
        return PI * radius * radius;  
    }  
    @Override
```

```

    public double perimeter() {
        return 2 * PI * radius;
    }
}

class Ellipse implements GeoAnalyzer {
    private double semiMajorAxis;
    private double semiMinorAxis;

    public Ellipse(double semiMajorAxis, double semiMinorAxis) {
        this.semiMajorAxis = semiMajorAxis;
        this.semiMinorAxis = semiMinorAxis;
    }

    @Override
    public double area() {
        return PI * semiMajorAxis * semiMinorAxis;
    }

    @Override
    public double perimeter() {
        // This is a rough approximation of the ellipse perimeter using Ramanujan's formula
        double a = semiMajorAxis;
        double b = semiMinorAxis;
        return PI * (3 * (a + b) - Math.sqrt((3 * a + b) * (a + 3 * b)));
    }
}

class Rectangle implements GeoAnalyzer {
    private double width;
    private double height;

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

```

```

    }
    @Override
    public double area() {
        return width * height;
    }
    @Override
    public double perimeter() {
        return 2 * (width + height);
    }
}

class Geometry {
    public static void run() {
        GeoAnalyzer circle = new Circle(5);
        System.out.println("Area of circle: " + circle.area());
        System.out.println("Perimeter of circle: " + circle.perimeter());

        GeoAnalyzer ellipse = new Ellipse(5, 3);
        System.out.println("Area of ellipse: " + ellipse.area());
        System.out.println("Perimeter of ellipse: " + ellipse.perimeter());
        GeoAnalyzer rectangle = new Rectangle(5, 7);
        System.out.println("Area of rectangle: " + rectangle.area());
        System.out.println("Perimeter of rectangle: " + rectangle.perimeter());
    }
}

```

OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java CP10
Area of circle: 78.53981633974483
Perimeter of circle: 31.41592653589793
Area of ellipse: 47.12388980384689
Perimeter of ellipse: 25.526986393758545
Area of rectangle: 35.0
Perimeter of rectangle: 24.0

```

**DP1:** Extend your Gravity Calculator code (Assignment AA1) to handle exceptions through a try-catch finally block. Handle provision for a divide by zero scenarios caught by NumberFormatException. Explicitly invoke this exception in execution and observe the response.

**SOLUTION:**

```
import java.util.Scanner;

public class DP1 {

    public static void main(String[] args) {

        double g = 9.81;

        Gravity earth = (double h, double r) -> {
            if (h + r == 0) {
                throw new ArithmeticException("Divide by zero");
            }
            return g * r * r / ((h + r) * (h + r));
        };

        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter the height: ");
            double height = sc.nextDouble();
            System.out.print("Enter the radius: ");
            double radius = sc.nextDouble();
            System.out.printf(
                "gravity at height h = %fs is %fm/s^2\n",
                height, earth.at(height, radius)
            );
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception:" + e.getMessage());
        }
    }
}
```

```

    } catch (Exception e) {
        System.out.println("Other Exception:" + e.getMessage());
    } finally {
        sc.close();
    }
}
}
}

```

@FunctionalInterface

```

interface Gravity {
    public double at(double a, double b)
        throws ArithmeticException;
}

```

OUTPUT-:



```

C:\Users\Aman Yadav\Desktop>java DP1
Enter the height: 112
Enter the radius: 25
gravity at height h = 112.000000s is 0.326669m/s^2

```

**DP2:** Further extend your menu-driven Simple Calculator code (Assignment 2) by adding support a Custom Exception code that raises an ‘Invalid Numeral’ exception each time the user tries to enter any character except a number for calculation. Explicitly invoke this exception in execution and observe the response.

**SOLUTION:**

```

import java.util.*;

public class DP2 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String ch;

        do{

            double num1=0;

            double num2=0;

            boolean check=false;

```

```

int choice;

try{
System.out.println("enter the first number");
num1=Double.parseDouble(sc.nextLine());
System.out.println("Enter the second number");
num2=Double.parseDouble(sc.nextLine());

}
catch(NumberFormatException e)
{
    System.out.println(e);
    check=true;
}
if(check==false)
{
    System.out.println("WELCOME TO THE CALCULATOR APPLICATION!!!!");
    System.out.println("SELECT FROM GIVEN OPTIONS");
    System.out.println("1.ADDITION OF TWO NUMBERS");
    System.out.println("2.SUBTRACTION OF 2 NUMBERS");
    System.out.println("3.DIVISION OF 2 NUMBERS");
    System.out.println("4.MULTIPLICATION OF 2 NUMBERS");
    System.out.println();
    System.out.println("ENTER YOUR CHOICE: ");
    choice=sc.nextInt();
    switch(choice)
    {
        case 1: System.out.println("THE ADDITION OF NUMBERS IS: "+(num1+num2));
                break;
        case 2: System.out.println("THE SUBTRACTION OF NUMBERS IS: "+(num1-num2));
                break;
        case 3: boolean flag=false;
                try{
                    System.out.println((int)(num1)/(int)(num2));

```

```

    }
    catch(ArithmeticException e)
    {
        System.out.println(e);
        flag=true;
    }
    if(flag==false)
    {
        System.out.println("THE DIVISION OF 2 NUMBERS IS: "+(num1/num2));
    }

        break;

    case 4: System.out.println("THE MULTIPLICATION OF 2 NUMBERS IS: "+(num1*num2));

        break;

    default: System.out.println("ENTER A VALID CHOICE");
}
}
sc.nextLine();

System.out.println("DO YOU WANT TO CONTINUE(Y/N)");
ch=sc.nextLine();
}while(ch.equals("y"));
sc.close();

}
}

```

OUTPUT-:



```
enter the first number
12
Enter the second number
11
WELCOME TO THE CALCULATOR APPLICATION!!!!
SELECT FROM GIVEN OPTIONS
1.ADDITION OF TWO NUMBERS
2.SUBTRACTION OF 2 NUMBERS
3.DIVISION OF 2 NUMBERS
4.MULTIPLICATION OF 2 NUMBERS

ENTER YOUR CHOICE:
1
THE ADDITION OF  NUMBERS IS: 23.0
DO YOU WANT TO CONTINUE(Y/N)
y
enter the first number
2
Enter the second number
3
WELCOME TO THE CALCULATOR APPLICATION!!!!
SELECT FROM GIVEN OPTIONS
1.ADDITION OF TWO NUMBERS
2.SUBTRACTION OF 2 NUMBERS
3.DIVISION OF 2 NUMBERS
4.MULTIPLICATION OF 2 NUMBERS

ENTER YOUR CHOICE:
3
0
THE DIVISION OF 2 NUMBERS IS: 0.6666666666666666
DO YOU WANT TO CONTINUE(Y/N)
y
```

```

enter the first number
1
Enter the second number
2
WELCOME TO THE CALCULATOR APPLICATION!!!!
SELECT FROM GIVEN OPTIONS
1.ADDITION OF TWO NUMBERS
2.SUBTRACTION OF 2 NUMBERS
3.DIVISION OF 2 NUMBERS
4.MULTIPLICATION OF 2 NUMBERS

ENTER YOUR CHOICE:
2
THE SUBTRACTION OF NUMBERS IS: -1.0
DO YOU WANT TO CONTINUE(Y/N)
y
enter the first number
2
Enter the second number
3
WELCOME TO THE CALCULATOR APPLICATION!!!!
SELECT FROM GIVEN OPTIONS
1.ADDITION OF TWO NUMBERS
2.SUBTRACTION OF 2 NUMBERS
3.DIVISION OF 2 NUMBERS
4.MULTIPLICATION OF 2 NUMBERS

ENTER YOUR CHOICE:
3
0
THE DIVISION OF 2 NUMBERS IS: 0.6666666666666666
DO YOU WANT TO CONTINUE(Y/N)
n

C:\Users\Aman Yadav\Desktop>

```

**DP3:** Create a class StudentRegistrationCheck that checks eligibility of a student for registration. If the student age<12 and marks<200 then the student is not eligible for registration.

**SOLUTION:**

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

```
public class DP3 {
    public static void main(String[] args) {
```

```
        StudentRegistrationCheck.run();
    }
}
```

```
class StudentRegistrationCheck {
    public static void run() {
        Scanner sc = new Scanner(System.in);
        int age = 0;
        int marks = 0;
        try {
            System.out.print("Enter student age: ");
            age = sc.nextInt();
            System.out.print("Enter student marks: ");
            marks = sc.nextInt();
            if (age < 12 || marks < 200) {
                throw new EligibilityException(
                    "Student is not eligible for registration"
                );
            }
            System.out.println("Student is eligible for registration");
        } catch (EligibilityException e) {
            System.out.println(e.getMessage());
        } finally {
            sc.close();
        }
    }
}
```

```
class EligibilityException extends Exception {
    public EligibilityException(String message) {
```

```
        super(message);  
    }  
}
```

OUTPUT-:

```
C:\Users\Aman Yadav\Desktop>java DP3  
Enter student age: 30  
Enter student marks: 56  
Student is not eligible for registration
```

**DP4:** An ExceptionPOC class is requesting a number between 1 and 10. Run the program again and enter 5.5. Although this number is between 1 and 10, the program will abort. Examine the error message. You should see the word Exception, the method where the exception occurred (main), the class name of the exception (InputMismatchException), as well as the call stack listing the method calls. Add a try/catch block to catch and handle the InputMismatchException exception. Identify the statements that cause the error as well as the portions of the program that depend upon these statements. Enclose these statements within the try block. Follow the try block with the catch block given below. Note, the InputMismatchException class is defined in java.util and must be imported. Also, when the Scanner throws an InputMismatchException, the input token will remain in the buffer so that it can be examined by the program. Complete code by implementing the same using:-

- a) Throws method declaration
- b) Throw keyword)

**SOLUTION:**

```
import java.util.InputMismatchException;  
import java.util.Scanner;  
  
public class DP4 {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        // Try block to handle InputMismatchException  
        try {  
            take_input(sc);  
        } catch (InputMismatchException e) {  
            // Catch block to handle InputMismatchException  
            System.out.println(  

```

```

        "InputMismatchException occurred: " + e.getMessage()
    );
} finally {
    sc.close();
}
}

static void take_input(Scanner sc) throws InputMismatchException {
    int number = 0;
    System.out.print("Enter a number between 1 and 10: ");
    number = sc.nextInt();
    if (number < 1 || number > 10) {
        throw new InputMismatchException(
            "Number is not within the range of 1 and 10"
        );
    }
    System.out.println("You entered: " + number);
}
}

```

OUTPUT-:

```

C:\Users\Aman Yadav\Desktop>java DP4
Enter a number between 1 and 10: 3
You entered: 3

C:\Users\Aman Yadav\Desktop>java DP4
Enter a number between 1 and 10: 0
InputMismatchException occurred: Number is not within the range of 1 and 10

```

**DP5:** Extend CP5 (Bank application) with appropriate exception handling.

**SOLUTION:**

```

import java.util.HashMap;
import java.util.Scanner;

```

```
public class DP5 {  
    static HashMap<Integer, BankAccount> accounts =  
        new HashMap<Integer, BankAccount>();  
  
    public static void addAccount(int accountNumber, int type) {  
        switch (type) {  
            case 1:  
                accounts.put(  
                    accountNumber,  
                    new SavingsAccount(accountNumber, 0)  
                );  
                break;  
            case 2:  
                accounts.put(  
                    accountNumber,  
                    new CurrentAccount(accountNumber, 0)  
                );  
                break;  
        }  
    }  
}  
  
public static void main(String[] args) {  
    int an = 1001;  
    String menu = "\n" +  
        "1. Create a new bank account \n" +  
        "2. Add money to the account \n" +  
        "3. Withdraw money from account \n" +  
        "4. Transfer money to bank account \n" +  
        "5. Get the balance \n" +  
        "6. Exit \n" +  
        "Enter your choice: ";
```

```
String menu2 =
    "Type of bank account: \n" +
    "  1. Savings account \n" +
    "  2. Current account \n" +
    "Enter your choice: ";
Scanner sc = new Scanner(System.in);
boolean isRunning = true;
while (isRunning) {
    System.out.print(menu);
    int choice = sc.nextInt();
    switch (choice) {
        case 1: {
            System.out.print(menu2);
            int type = sc.nextInt();
            addAccount(an, type);
            System.out.println("Your account no is " + an++);
            break;
        }
        case 2: {
            System.out.print("Enter the account_no: ");
            int account_number = sc.nextInt();
            System.out.print("Enter the amount: ");
            float amount = sc.nextFloat();
            try {
                if (amount < 0) {
                    throw new InsufficientFundsException(
                        "Invalid amount:" +
                        " cannot deposit negative amount"
                    );
                } else {
```

```

        if (!accounts.containsKey(account_number)) {
            throw new InvalidAccountException(
                "Invalid account number"
            );
        }
        accounts.get(account_number).deposit(amount);
    }
} catch (InsufficientFundsException e) {
    System.out.println(e.getMessage());
} catch (InvalidAccountException e) {
    System.out.println(e.getMessage());
}

break;
}
case 3: {
    System.out.print("Enter the account_no: ");
    int account_number = sc.nextInt();
    System.out.print("Enter the amount: ");
    float amount = sc.nextFloat();
    try {
        if (!accounts.containsKey(account_number))
            throw new InvalidAccountException(
                "Invalid account number"
            );
        if (!accounts.get(account_number).withdraw(amount))
            throw new InsufficientFundsException(
                "Low Balance"
            );
    } catch (

```



```

        InsufficientFundsException |
        InvalidAccountException e) {
            System.out.println(e.getMessage());
        }
        break;
    }
case 4: {
    System.out.print("Enter the account_no: ");
    int account_number = sc.nextInt();
    System.out.print("Enter 2nd account_no: ");
    int r_account_number = sc.nextInt();
    System.out.print("Enter the amount: ");
    float amount = sc.nextFloat();
    try {
        if (!accounts.containsKey(r_account_number)) {
            throw new InvalidAccountException(
                "Invalid account number"
            );
        }
        if (!accounts.get(account_number)
            .transfer(
                accounts.get(r_account_number),
                amount
            ))
            throw new InsufficientFundsException(
                "Low Balance"
            );
    } catch (InsufficientFundsException e) {
        System.out.println(e.getMessage());
    } catch (InvalidAccountException e) {

```

```

        System.out.println(e.getMessage());
    }
    break;
}
case 5: {
    System.out.print("Enter the account_no: ");
    int account_number = sc.nextInt();
    try {
        if (!accounts.containsKey(account_number)) {
            throw new InvalidAccountException(
                "Invalid account number"
            );
        } else {
            System.out.println(
                "balance: " +
                accounts.get(account_number).getBalance()
            );
        }
    } catch (InvalidAccountException e) {
        System.out.println(e.getMessage());
    }
    break;
}
case 6: {
    sc.close();
    isRunning = false;
    break;
}
}
}

```

```
}  
}
```

```
class BankAccount {  
    int account_id;  
    float balance;  
  
    public BankAccount(int account_id, float balance) {  
        this.account_id = account_id;  
        this.balance = balance;  
    }  
  
    public float getBalance() {  
        return balance;  
    }  
  
    public float getAccountId() {  
        return account_id;  
    }  
  
    public void deposit(float amount) {  
        System.out.println("Depositing: " + amount +  
            " In account " + account_id);  
        balance += amount;  
    }  
  
    public boolean withdraw(float amount) {  
        if (amount > balance) {  
            System.out.println("Low balance: "  
                + balance +
```

```

        " Can't withdraw");
    return false;
}
System.out.println("withdrawing: " + amount +
    " From: " + account_id);
balance -= amount;
return true;
}

```

```

public boolean transfer(BankAccount account, float amount) {
    if (amount > balance) {
        System.out.println("Low balance: " + balance + " Can't transfer");
        return false;
    }
    System.out.println("transferring: " + amount +
        "\nFrom: " + account_id +
        "\nTo: " + account.account_id);
    account.balance += amount;
    balance -= amount;
    return true;
}
}

```

```

class SavingsAccount extends BankAccount {
    static int max_transactions = 10, max_withdraws = 10;
    int transactions = 0;
    int withdraws = 0;

    SavingsAccount(int account_id, float balance) {
        super(account_id, balance);
    }
}

```

```
}
```

```
@Override
```

```
public boolean withdraw(float amount) {  
    if (withdraws == max_withdraws) {  
        System.out.println("number of withdraws exceeded");  
        return false;  
    }  
    withdraws++;  
    return super.withdraw(amount);  
}
```

```
@Override
```

```
public boolean transfer(BankAccount account, float amount) {  
    if (transactions == max_transactions) {  
        System.out.println("number of transfers exceeded");  
        return false;  
    }  
    transactions++;  
    return super.transfer(account, amount);  
}  
}
```

```
class CurrentAccount extends BankAccount {  
    CurrentAccount(int account_id, float balance) {  
        super(account_id, balance);  
    }  
}
```

```
class InsufficientFundsException extends Exception {
```

```
public InsufficientFundsException(String message) {  
    super(message);  
}  
}
```

```
class InvalidAccountException extends Exception {  
    public InvalidAccountException(String message) {  
        super(message);  
    }  
}
```

OUTPUT-:

```
C:\Users\Aman Yadav\Desktop>java DP5
```

```
1. Create a new bank account  
2. Add money to the account  
3. Withdraw money from account  
4. Transfer money to bank account  
5. Get the balance  
6. Exit
```

```
Enter your choice: 1
```

```
Type of bank account:
```

- 1. Savings account
- 2. Current account

```
Enter your choice: 1
```

```
Your account no is 1001
```

```
1. Create a new bank account  
2. Add money to the account  
3. Withdraw money from account  
4. Transfer money to bank account  
5. Get the balance  
6. Exit
```

```
Enter your choice: 2
```

```
Enter the account_no: 1001
```

```
Enter the amount: 40000
```

```
Depositing: 40000.0 In account 1001
```

```
1. Create a new bank account  
2. Add money to the account  
3. Withdraw money from account  
4. Transfer money to bank account  
5. Get the balance  
6. Exit
```

```
Enter your choice: 6
```

**DP6:** Create an Exception class InvalidProductException that can be thrown if a user adds an invalid product.

**SOLUTION:**

```
import java.util.ArrayList;
import java.util.List;

public class DP6 {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
        try {
            products.add(new Product("Product 1", 10));
            products.add(new Product("Product 2", 20));
            // This will throw an InvalidProductException
            products.add(new Product("", 30));
        } catch (InvalidProductException e) {
            System.out.println(e.getMessage());
        }
        for (Product product : products) {
            System.out.println(product.getName() + ": " + product.getPrice());
        }
    }
}

class Product {
    private String name;
    private int price;

    public Product(String name, int price) throws InvalidProductException {
```

```
    if (name == null || name.isEmpty()) {  
        throw new InvalidProductException("Invalid product name");  
    }  
    if (price <= 0) {  
        throw new InvalidProductException("Invalid product price");  
    }  
    this.name = name;  
    this.price = price;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public int getPrice() {  
    return price;  
}  
}
```

```
class InvalidProductException extends Exception {  
    public InvalidProductException() {  
        super();  
    }  
}
```

```
public InvalidProductException(String message) {  
    super(message);  
}
```

```
public InvalidProductException(String message, Throwable cause) {  
    super(message, cause);  
}
```

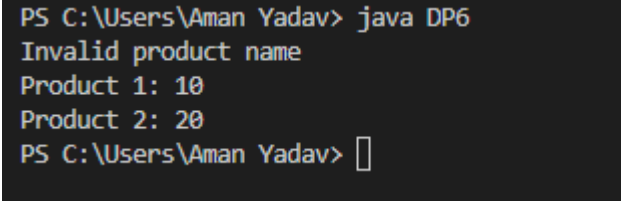


```
}
```

```
public InvalidProductException(Throwable cause) {  
    super(cause);  
}
```

```
}
```

## OUTPUT-:



```
PS C:\Users\Aman Yadav> java DP6  
Invalid product name  
Product 1: 10  
Product 2: 20  
PS C:\Users\Aman Yadav> 
```

**DP7:** Compile and run BadThreads.java:

## SOLUTION:

```
import java.util.concurrent.atomic.AtomicReference;
```

```
public class DP7 {  
    public static void main(String[] args) {  
        try {  
            BadThreads.run();  
        } catch (InterruptedException e) {  
            System.out.println(e);  
        }  
    }  
}
```

```
class BadThreads {  
    static String message1;  
    static volatile String message2;
```

```
static AtomicReference<String> message3 = new AtomicReference<>();
```

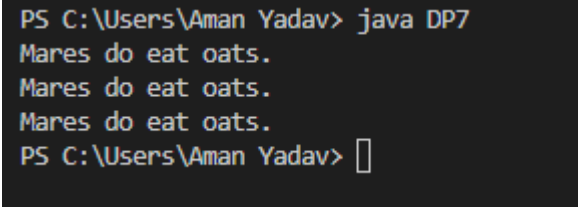
```
private static class CorrectorThread extends Thread {  
    public void run() {  
        try {  
            sleep(1000);  
        } catch (InterruptedException e) {  
        }  
        // Key statement 1:  
        synchronized (BadThreads.class) {  
            message1 = "Mares do eat oats.";  
        }  
        message2 = "Mares do eat oats.";  
        message3.set("Mares do eat oats.");  
    }  
}
```

```
public static void run() throws InterruptedException {  
    (new CorrectorThread()).start();  
    synchronized (BadThreads.class) {  
        message1 = "Mares do not eat oats.";  
    }  
    message3.set("Mares do not eat oats.");  
    Thread.sleep(2000);  
    // Key statement 2:  
    synchronized (BadThreads.class) {  
        System.out.println(message1);  
    }  
    System.out.println(message2);  
    System.out.println(message3.get());  
}
```

```
}
```

```
}
```

## OUTPUT-:



```
PS C:\Users\Aman Yadav> java DP7
Mares do eat oats.
Mares do eat oats.
Mares do eat oats.
PS C:\Users\Aman Yadav> 
```

**DP8:** Implement the producer consumer problem using multithreading in java. In computing, the producer–consumer problem (also known as the bounded- buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

- a) The producer’s job is to generate data, put it into the buffer, and start again.
- b) At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time. To make sure that the producer won’t try to add data into the buffer if it’s full and that the consumer won’t try to remove data from an empty buffer.

## SOLUTION:

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

```
public class DP8 {
```

```
    public static void main(String[] args) {
```

```
        Queue<Integer> product_queue = new Queue<Integer>(5);
```

```
        Producer p = new Producer(product_queue);
```

```
        Consumer c = new Consumer(product_queue);
```

```
        p.start();
```

```
        c.start();
```

```
    }
```

```
}
```

```

class Queue<T> extends LinkedList<T> {
    private int qsize;
    Queue (int size) {
        this.qsize = size;
    }

    public int qSize() {
        return qsize;
    }

    @Override
    public boolean add(T o) {
        while (this.size() == this.qsize) {
            return false;
        }
        super.add(o);
        return true;
    }
}

```

```

class Producer extends Thread {
    private Queue<Integer> queue;

    Producer (Queue<Integer> queue) {
        this.queue = queue;
    }

    synchronized boolean add(Integer i) {
        System.out.println("Producer: " + i);
        return queue.add(i);
    }
}

```

```
}
```

```
@Override
```

```
public void run() {
```

```
    int i = 0;
```

```
    while (i < 10) {
```

```
        if (add(i))
```

```
            i++;
```

```
        try {
```

```
            Thread.sleep(10);
```

```
        }
```

```
        catch (Exception e) {
```

```
            System.err.println(e);
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
class Consumer extends Thread {
```

```
    private Queue<Integer> queue;
```

```
    int consumed_no;
```

```
    Consumer (Queue<Integer> queue) {
```

```
        this.queue = queue;
```

```
        this.consumed_no = 0;
```

```
    }
```

```
    synchronized void consume() {
```

```
        System.out.println("Consumer: " + queue.remove());
```

```
        this.consumed_no += 1;
```

```
}
```

```
@Override
```

```
public void run() {
```

```
    while(consumed_no < 10) {
```

```
        if(queue.size() != 0) consume();
```

```
        try {
```

```
            Thread.sleep(10);
```

```
        }
```

```
        catch (Exception e) {
```

```
            System.err.println(e);
```

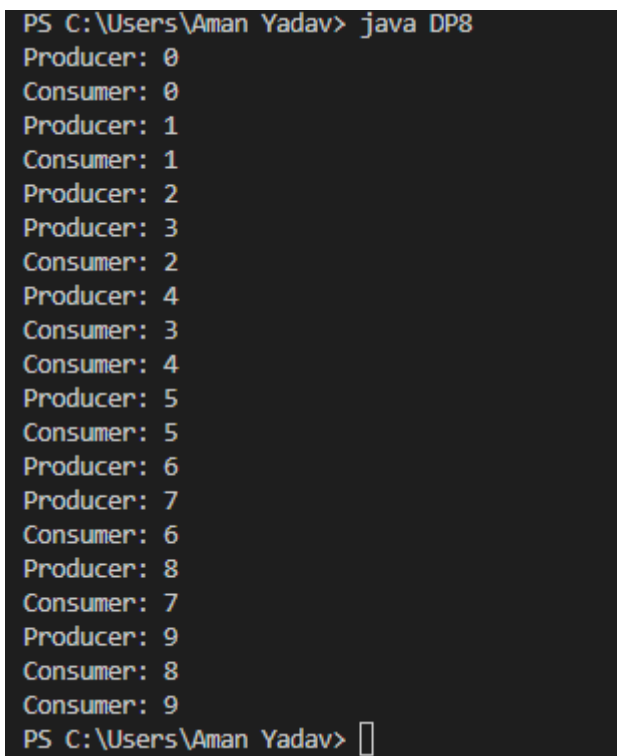
```
        }
```

```
    }
```

```
}
```

```
}
```

## OUTPUT:-



```
PS C:\Users\Aman Yadav> java DP8
Producer: 0
Consumer: 0
Producer: 1
Consumer: 1
Producer: 2
Producer: 3
Consumer: 2
Producer: 4
Consumer: 3
Consumer: 4
Producer: 5
Consumer: 5
Producer: 6
Producer: 7
Consumer: 6
Producer: 8
Consumer: 7
Producer: 9
Consumer: 8
Consumer: 9
PS C:\Users\Aman Yadav>
```

**DP9:** Create a Java application that executes concurrent transactions in a bank.

## SOLUTION:

```

public class DP9 {
    public static void main(String[] args) {
        BankAccount a1 = new BankAccount(1, 10000.0f);

        Thread t1 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10; i++) {
                    double amount = Math.round(Math.random() * 10000);
                    if (a1.removeMoney(amount)) {
                        System.out.println("Successfully removed: " + amount);
                    } else {
                        System.out.println("Low Balance");
                    }
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            public void run() {
                for(int i = 0; i < 10; i++) {
                    double amount = Math.round(Math.random() * 10000);
                    a1.addMoney(amount);
                    System.out.println("Added: " + amount);
                }
            }
        });

        t1.start();
        t2.start();
    }
}

```

```

    }
}

class BankAccount {
    int account_id;
    double balance;

    BankAccount(int account_id, double balance) {
        this.account_id = account_id;
        this.balance = balance;
    }

    public synchronized void addMoney(double amount) {
        balance += amount;
    }

    public synchronized boolean removeMoney(double amount) {
        if (balance - amount < 0)
            return false;
        balance -= amount;
        return true;
    }

    public void getBalance() {
        System.out.println("balance: " + balance);
    }
}

```

**OUTPUT-:**



```
PS C:\Users\Aman Yadav> javac DP9.java
PS C:\Users\Aman Yadav> java DP9
Successfully removed: 2436.0
Added: 7203.0
Successfully removed: 7837.0
Successfully removed: 2745.0
Added: 6730.0
Added: 2126.0
Successfully removed: 5452.0
Added: 6117.0
Successfully removed: 8210.0
Added: 6453.0
Successfully removed: 3969.0
Successfully removed: 3180.0
Successfully removed: 5008.0
Added: 8068.0
Successfully removed: 7548.0
Added: 8181.0
Successfully removed: 4670.0
Added: 3976.0
Added: 6091.0
Added: 8454.0
PS C:\Users\Aman Yadav> 
```

**DP10:** Create a list of numbers and then sort in ascending order as well as in descending order simultaneously

#### **SOLUTION:**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class DP10 {
    public static void main(String[] args) {
        // Create an ArrayList of numbers
        List<Integer> numbers = new ArrayList<>();
        numbers.add(5);
        numbers.add(3);
        numbers.add(8);
        numbers.add(1);
        numbers.add(9);
        numbers.add(4);
```

```
// Create a thread to sort the list in ascending order
Thread ascendingThread = new SortThread(numbers, true);

// Create a thread to sort the list in descending order
Thread descendingThread = new SortThread(numbers, false);


// Start both threads
ascendingThread.start();

try {
    ascendingThread.join();
}
catch (InterruptedException e) {
    System.out.println(e);
}

descendingThread.start();
}
}
```

```
class SortThread extends Thread {
    private List<Integer> list;
    private boolean ascending;

    public SortThread(List<Integer> list, boolean ascending) {
        this.list = list;
        this.ascending = ascending;
    }

    public synchronized void sort() {
        if (ascending) {
            Collections.sort(list);
        } else {
            Collections.sort(list, Collections.reverseOrder());
        }
    }
}
```

```

@Override

public void run() {

    sort();

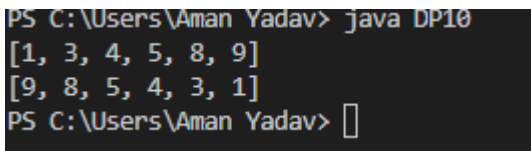
    System.out.println(list);

}

}

```

## OUTPUT-:



```

PS C:\Users\Aman Yadav> java DP10
[1, 3, 4, 5, 8, 9]
[9, 8, 5, 4, 3, 1]
PS C:\Users\Aman Yadav> 

```

**EP1:** Model Person with name and age. Manage instances of Person by ensuring that no two instances are duplicated?

## SOLUTION:

```
import java.util.HashSet;
```

```

public class EP1 {

    public static void main(String[] args) {

        Person p1 = new Person("john", 12);
        Person p2 = new Person("john", 12);
        Person p3 = new Person("doe", 12);
        Person p4 = new Person("john", 13);
        Person p5 = new Person("Mark", 15);

        HashSet<Person> persons = new HashSet<Person>();
        persons.add(p1);
        persons.add(p2);
        persons.add(p3);
        persons.add(p4);
        persons.add(p5);
    }
}

```

```
        System.out.println(persons);
    }
}
```

```
class Person {
    public String Name;
    public int Age;

    public Person(String name, int age) {
        Name = name;
        Age = age;
    }

    @Override
    public String toString() {
        return String.format("%s is %d years old", Name, Age);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Person other = (Person) obj;
        if (this.Age == other.Age && this.Name == other.Name)
            return true;
        return false;
    }
}
```

```
}
```

```
@Override
```

```
public int hashCode() {
```

```
    return this.Age + this.Name.hashCode();
```

```
}
```

```
}
```

## OUTPUT-:

```
PS C:\Users\Aman Yadav> java EP1
[john is 12 years old, doe is 12 years old, Mark is 15 years old, john is 13 years old]
PS C:\Users\Aman Yadav>
```

**EP2:** Create a subclass of Person (in EP1 above), called ComparablePerson which implements Comparable<Person> interface, and try out the Collections.sort() and Collections.binarySearch() methods on the same.

## SOLUTION:

```
import java.util.HashSet;
```

```
import java.util.ArrayList;
```

```
import java.util.Collections;
```

```
public class EP2 {
```

```
    public static void main(String[] args) {
```

```
        ComparablePerson cp1 = new ComparablePerson("john", 12);
```

```
        ComparablePerson cp2 = new ComparablePerson("doe", 12);
```

```
        ComparablePerson cp3 = new ComparablePerson("john", 13);
```

```
        ComparablePerson cp4 = new ComparablePerson("mark", 15);
```

```
        ArrayList<ComparablePerson> cpersons =
```

```
            new ArrayList<ComparablePerson>();
```

```
        cpersons.add(cp1);
```

```
        cpersons.add(cp2);
```

```
cpersons.add(cp3);
```

```
cpersons.add(cp4);
```

```
Collections.sort(cpersons);
```

```
System.out.println(cpersons);
```

```
System.out.println(
```

```
    "Finding John 12: " +
```

```
    Collections.binarySearch(
```

```
        cpersons,
```

```
        new ComparablePerson("john", 12)));
```

```
    }
```

```
}
```

```
class Person {
```

```
    public String Name;
```

```
    public int Age;
```

```
    public Person(String name, int age) {
```

```
        Name = name;
```

```
        Age = age;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return String.format("%s is %d years old", Name, Age);
```

```
    }
```

```
    @Override
```

```

public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Person other = (Person) obj;
    if (this.Age == other.Age && this.Name == other.Name)
        return true;
    return false;
}

```

@Override

```

public int hashCode() {
    return this.Age ^ this.Name.hashCode();
}
}

```

class ComparablePerson extends Person implements Comparable<ComparablePerson> {

```

    public ComparablePerson(String name, int age) {
        super(name, age);
    }

```

@Override

```

public int compareTo(ComparablePerson o) {
    return this.toString().compareTo(o.toString());
}
}

```

**OUTPUT-:**

```
PS C:\Users\Aman Yadav> java EP2
[doe is 12 years old, john is 12 years old, john is 13 years old, mark is 15 years old]
Finding John 12: 1
PS C:\Users\Aman Yadav> █
```

**EP3:** Model AddressBookEntry that prints name,address and phone of a person. Allow comparison of AddressBookEntry to compare name in a case sensitive manner?

### **SOLUTION:**

```
import java.util.Comparator;
import java.util.ArrayList;
import java.util.Collections;
```

```
public class EP3 {
    public static void main(String[] args) {
        ArrayList<AddressBookEntry> entries = new ArrayList<>();
        entries.add(new AddressBookEntry("Alice", "123 Main St", "555-1212"));
        entries.add(new AddressBookEntry("Bob", "456 Oak Ave", "555-1212"));
        entries.add(new AddressBookEntry("Charlie", "789 Pine St", "555-1212"));

        Collections.sort(entries);

        for (AddressBookEntry entry : entries) {
            entry.print();
        }
    }
}
```

```
class AddressBookEntry implements Comparable<AddressBookEntry> {
    private String name;
    private String address;
```



```
private String phone;
```

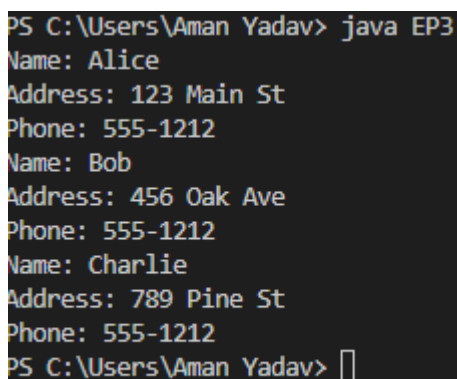
```
public AddressBookEntry(String name, String address, String phone) {  
    this.name = name;  
    this.address = address;  
    this.phone = phone;  
}
```

```
public void print() {  
    System.out.println("Name: " + name);  
    System.out.println("Address: " + address);  
    System.out.println("Phone: " + phone);  
}
```

@Override

```
public int compareTo(AddressBookEntry other) {  
    return this.name.compareTo(other.name);  
}  
}
```

## OUTPUT-:



```
PS C:\Users\Aman Yadav> java EP3  
Name: Alice  
Address: 123 Main St  
Phone: 555-1212  
Name: Bob  
Address: 456 Oak Ave  
Phone: 555-1212  
Name: Charlie  
Address: 789 Pine St  
Phone: 555-1212  
PS C:\Users\Aman Yadav> █
```

**EP4:** Counts the frequency of each of the words in a file given in the command-line, and save in a map of {word, freq}.

## SOLUTION:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
```

```
public class EP4 {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: EP4 <file>");
            System.exit(1);
        }
        try {
            WordFrequencyCounter.run(args[0]);
        }
        catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
class WordFrequencyCounter {
    public static void run(String fileName) throws IOException {
        Map<String, Integer> wordCounts = new HashMap<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line = null;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+");
```

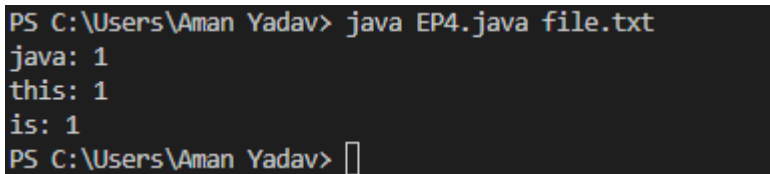
```

    for (String word : words) {
        word = word.toLowerCase();
        if (wordCounts.containsKey(word)) {
            wordCounts.put(word, wordCounts.get(word) + 1);
        } else {
            wordCounts.put(word, 1);
        }
    }
}

for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {
    System.out.printf("%s: %d\n", entry.getKey(), entry.getValue());
}
}
}

```

## OUTPUT-:



```

PS C:\Users\Aman Yadav> java EP4.java file.txt
java: 1
this: 1
is: 1
PS C:\Users\Aman Yadav> 

```

## FP1-:

```

import java.net.InetAddress;
import java.net.UnknownHostException;

class FP1 {
    public static void main(String[] args) {
        InetAddress ip;
        String Hostname;
        try {
            ip = InetAddress.getLocalHost();
            Hostname = ip.getHostName();

```

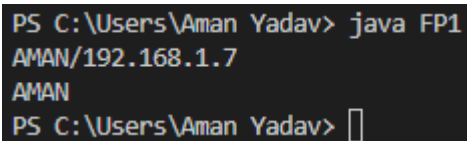
```

        println(ip);
        println(Hostname);
    } catch (UnknownHostException e) {
        println("UnknownHostException " + e);
    }
}

public static void println(Object line) {
    System.out.println(line);
}
}

```

## OUTPUT:-



```

PS C:\Users\Aman Yadav> java FP1
AMAN/192.168.1.7
AMAN
PS C:\Users\Aman Yadav> 

```

## FP2:- Create a solution to understand the different components of a URL?

// SOLN:-

```

import java.util.*;
//importing UL class
import java.net.URL;

public class FP2
{
    public static void main(String[] args) throws Exception
    {
        URL url = new URL("https://leetcode.com/");
        System.out.println("URL is : "+ url.toString());
        System.out.println("protocol is: "+ url.getProtocol());
        System.out.println("file name is: "+ url.getFile());
        System.out.println("host is: " + url.getHost());
        System.out.println("path is: " + url.getPath());
        System.out.println("port is: " + url.getPort());
        System.out.println("default port is: "+ url.getDefaultPort());
    }
}

```

```

PS C:\Users\Aman Yadav> javac FP2.java
PS C:\Users\Aman Yadav> java FP2
URL is : https://leetcode.com/
protocol is: https
file name is: /
host is: leetcode.com
path is: /
port is: -1
default port is: 443

```

FP3 Create a connection-less client/server application using UDP protocol that sends system date and time in the format requested by the client.

a. Client: Reads a string representing the required format from the end-user

b. Server: returns the system date and time in the requested format or a default format if received format is not understandable

//Client code

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class Client {

    public static void main(String[] args) throws Exception {

        // Read the required format from the end-user
        BufferedReader reader =
            new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Enter date and time format: ");

        String format = reader.readLine();

        // Create a datagram socket and send the format to the server
        DatagramSocket socket = new DatagramSocket();

        byte[] data = format.getBytes();

        InetAddress address = InetAddress.getByName("localhost");

        int port = 5000;

        DatagramPacket packet =
            new DatagramPacket(data, data.length, address, port);

        socket.send(packet);

        // Receive the date and time from the server
    }
}

```

```

byte[] buffer = new byte[1024];
packet = new DatagramPacket(buffer, buffer.length);
socket.receive(packet);
String dateTime = new String(packet.getData(), 0,
packet.getLength());
System.out.println("Date and time: " + dateTime);
}
}

```

//server code

```

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Server {
    public static void main(String[] args) throws Exception {
        // Create a datagram socket and wait for requests
        DatagramSocket socket = new DatagramSocket(5000);
        while (true) {
            // Receive the format from the client
            byte[] buffer = new byte[1024];
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
            socket.receive(packet);
            String format = new String(
                packet.getData(),
                0,
                packet.getLength());
            // Send the date and time to the client
            Date date = new Date();
            SimpleDateFormat sdf;
            try {
                // Use the requested format if it is valid
                sdf = new SimpleDateFormat(format);
            } catch (IllegalArgumentException e) {

```

```

        // Use a default format if the requested format is not
        sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    }
    String dateTime = sdf.format(date);
    byte[] data = dateTime.getBytes();
    packet = new DatagramPacket(
        data, data.length,
        packet.getAddress(), packet.getPort());

    socket.send(packet);
}
}
}

```

```

PS C:\Users\Aman Yadav> java Client.java
Enter date and time format: d M y
Date and time: 25 1 2023
PS C:\Users\Aman Yadav> 

```

**FP4:- Client:** Display the required contents. A connection-oriented client/server application using TCP/IP protocol where the client-server has the following responsibilities:

- a. **Server:** Creates an Employee class having fields like employeeName, employeeID, and department. Server holds an array of employee objects.
- b. **Client :** accepts the employeeID of an employee as an integer from the user.
- c. **Server:**Searches for the corresponding employee object, in the array and writes its details to the client stream.

**Client:** displays the received object's information.

```

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) throws Exception {
        // Read the employee ID from the user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter employee ID: ");
    }
}

```

```

int employeeID = scanner.nextInt();

// Connect to the server and send the employee ID
Socket socket = new Socket("localhost", 5000);
ObjectOutputStream output =
    new ObjectOutputStream(socket.getOutputStream());
output.writeInt(employeeID);
output.flush();

// Receive the employee object from the server
ObjectInputStream input =
    new ObjectInputStream(socket.getInputStream());
Employee employee = (Employee) input.readObject();

// Display the received object's information
if (employee != null) {
    System.out.println(employee);
} else {
    System.out.println("Employee not found");
}
}

class Employee implements java.io.Serializable {
    private String employeeName;
    private int employeeID;
    private String department;

    public Employee(String employeeName, int employeeID, String department) {
        this.employeeName = employeeName;
        this.employeeID = employeeID;
        this.department = department;
    }
}

```



```

    public String getEmployeeName() {
        return employeeName;
    }

    public int getEmployeeID() {
        return employeeID;
    }

    public String getDepartment() {
        return department;
    }

    @Override
    public String toString() {
        return
            "Employee Name: " + employeeName +
            ", Employee ID: " + employeeID +
            ", Department: " + department;
    }
}

//server code
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) throws Exception {
        // Create an array of employee objects
        Employee[] employees = {
            new Employee("John Smith", 12345, "IT"),

```

```

        new Employee("Jane Doe", 54321, "HR"),
        new Employee("Bob Johnson", 23456, "Sales")
    };

    // Create a server socket and wait for client connections
    ServerSocket serverSocket = new ServerSocket(5000);
    while (true) {
        Socket socket = serverSocket.accept();
        ObjectInputStream input =
            new ObjectInputStream(socket.getInputStream());
        int employeeID = input.readInt();
        Employee employee = null;
        for (Employee e : employees) {
            if (e.getEmployeeID() == employeeID) {
                employee = e;
                break;
            }
        }
        ObjectOutputStream output =
            new ObjectOutputStream(socket.getOutputStream());
        output.writeObject(employee);
        output.flush();
        socket.close();
    }
}

class Employee implements java.io.Serializable {
    private String employeeName;
    private int employeeID;
    private String department;

    public Employee(String employeeName, int employeeID, String department) {

```

```

        this.employeeName = employeeName;
        this.employeeID = employeeID;
        this.department = department;
    }

    public String getEmployeeName() {
        return employeeName;
    }

    public int getEmployeeID() {
        return employeeID;
    }

    public String getDepartment() {
        return department;
    }

    @Override
    public String toString() {
        return
            "Employee Name: " + employeeName +
            ", Employee ID: " + employeeID +
            ", Department: " + department;
    }
}

```

```

PS C:\Users\Aman Yadav> java Client
Enter employee ID: 12345
Employee Name: John Smith, Employee ID: 12345, Department: IT
PS C:\Users\Aman Yadav> 

```

**FP5:- Create a calculator based client/server application where the client sends request to the server in the form of an arithmetic equation of the form “operand1 operator operator2”. The server should respond back to answer the equation?**

//server

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) throws Exception {
        // Create a server socket and wait for client connections
        ServerSocket serverSocket = new ServerSocket(5000);
        while (true) {
            Socket socket = serverSocket.accept();
            BufferedReader input = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
            PrintWriter output = new PrintWriter(
                socket.getOutputStream(), true);
            String equation = input.readLine();
            double result = evaluate(equation);
            output.println(result);
            socket.close();
        }
    }

    public static double evaluate(String equation) {
        // Split the equation into its operands and operator
        String[] parts = equation.split(" ");
        double operand1 = Double.parseDouble(parts[0]);
        double operand2 = Double.parseDouble(parts[2]);
        String operator = parts[1];

        // Evaluate the equation based on the operator
        if (operator.equals("+")) {
            return operand1 + operand2;
        }
    }
}
```

```

        } else if (operator.equals("-")) {
            return operand1 - operand2;
        } else if (operator.equals("*")) {
            return operand1 * operand2;
        } else if (operator.equals("/")) {
            return operand1 / operand2;
        } else {
            throw new IllegalArgumentException(
                "Invalid operator: " + operator);
        }
    }
}

//client

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) throws Exception {
        // Read the equation from the user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter equation: ");
        String equation = scanner.nextLine();

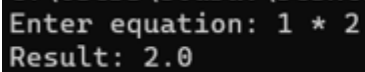
        // Connect to the server and send the equation
        Socket socket = new Socket("localhost", 5000);
        PrintWriter output = new PrintWriter(socket.getOutputStream(), true);
        output.println(equation);
        BufferedReader input = new BufferedReader(
            new InputStreamReader(socket.getInputStream()));
    }
}

```

```

        // Read the result from the server
        double result = Double.parseDouble(input.readLine());
        System.out.println("Result: " + result);
        socket.close();
        scanner.close();
    }
}

```



```

Enter equation: 1 * 2
Result: 2.0

```

// GP1:- create a class called fruits that has a method mango() that tells if the mango is sweet or sour. Suppose we need a sour mango in taste for only 1 time. Realize this temporary requirement through an anonymous inner class?

```

class GP1
{
    public static void main(String[] args)
    {
        Taste t1 = new Taste();
        t1.taste();
    }
}

class Fruits
{
    void mango()
    {
        System.out.println("Mango is sweet");
    }
    void apple()
    {
        System.out.println("Apple is sweet");
    }
    void orange()
    {
        System.out.println("Orange is sour");
    }
}

```

```

    }
    void grapes()
    {
        System.out.println("Grapes are sour");
    }
}
class Taste
{
    void taste()
    {
        Fruits f1 = new Fruits(){
            void mango()
            {
                System.out.println("Mango is sour");
            }
        };
        f1.mango();
    }
}

```

```

PS C:\Users\Aman Yadav> java GP1
Mango is sour
PS C:\Users\Aman Yadav> 

```

// GP2:- Create a class AWTCounter that starts a counter from 0 and increments its value on every button click?

// SOL:-

```

import java.awt.*;           // Using AWT container and component classes
import java.awt.event.*;    // Using AWT event classes and listener interfaces

```

// An AWT program inherits from the top-level container java.awt.Frame

```

public class GP2 extends Frame {
    private Label lblCount;    // Declare a Label component
    private TextField tfCount; // Declare a TextField component
    private Button btnCount;   // Declare a Button component
    private int count = 0;     // Counter's value

```

```

// Constructor to setup GUI components and event handlers
public GP2 () {
    setLayout(new FlowLayout());

    // "super" Frame, which is a Container, sets its layout to FlowLayout to
    arrange
    // the components from left-to-right, and flow to next row from top-to-
    bottom.

    lblCount = new Label("Counter"); // construct the Label component
    add(lblCount);                    // "super" Frame container adds Label
    component

    tfCount = new TextField(count + "", 10); // construct the TextField
    component with initial text
    tfCount.setEditable(false);           // set to read-only
    add(tfCount);                         // "super" Frame container adds TextField
    component

    btnCount = new Button("Count"); // construct the Button component
    add(btnCount);                  // "super" Frame container adds Button
    component

    BtnCountListener listener = new BtnCountListener();
    btnCount.addActionListener(listener);

    // "btnCount" is the source object that fires an(ActionEvent when
    clicked.

    // The source object adds an instance of BtnCountListener as an
   (ActionEvent listener,
    // which provides an(ActionEvent handler called actionPerformed().
    // Clicking "Count" button calls back actionPerformed().

    setTitle("AWT Counter"); // "super" Frame sets its title
    setSize(300, 100);       // "super" Frame sets its initial window size

    // For inspecting the Container/Components objects

```



```

// System.out.println(this);
// System.out.println(lblCount);
// System.out.println(tfCount);
// System.out.println(btnCount);
setVisible(true);          // "super" Frame shows
// System.out.println(this);
// System.out.println(lblCount);
// System.out.println(tfCount);
// System.out.println(btnCount);
}

```

```

// The entry main() method
public static void main(String[] args) {
    // Invoke the constructor to setup the GUI, by allocating an instance
    GP2 app = new GP2();
    // or simply "new AWTCounter();" for an anonymous instance
}

```

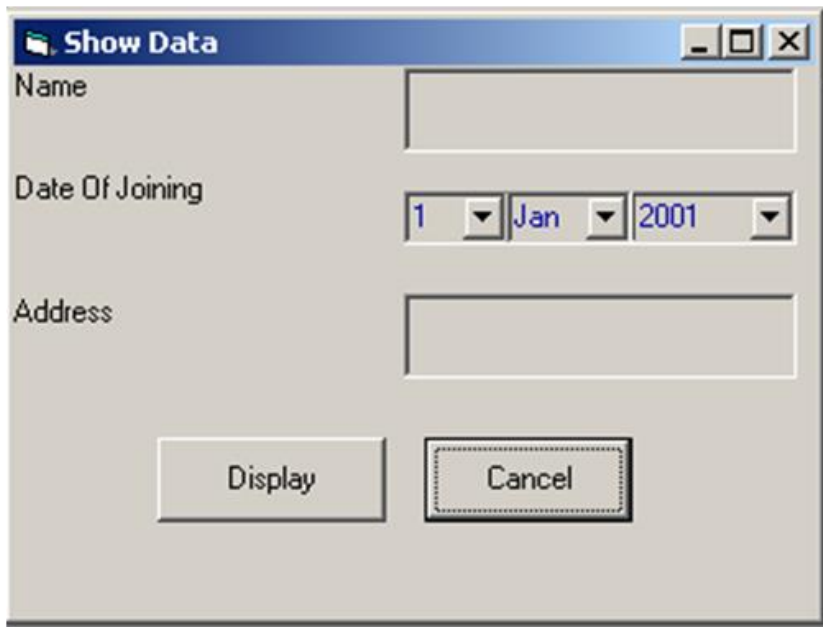
```

// Define an inner class to handle the "Count" button-click
private class BtnCountListener implements ActionListener {
    // ActionEvent handler - Called back upon button-click.
    @Override
    public void actionPerformed(ActionEvent evt) {
        ++count; // Increase the counter value
        // Display the counter value on the TextField tfCount
        tfCount.setText(count + ""); // Convert integer to String
    }
}
}
}

```



GP3:- Create the following layout using awt/swing



When user clicks the “Display” button the data entered by the user should be displayed in another frame window.

When user clicks the “Cancel” button the data fields should get cleared.

```
import java.awt.*;
import java.awt.event.*;

public class GP3_AwtForm extends Frame implements ActionListener{
    public static void main(String[] args) {
        GP3_AwtForm awt = new GP3_AwtForm();
    }
    TextField firstNameTF;
    TextField lastNameTF;
    TextField dobTF;

    GP3_AwtForm(){
```

```
setSize(300,300);
setLayout(null);
setVisible(true);
setTitle("AWT Form");

addWindowListener (new WindowAdapter() {
    public void windowClosing (WindowEvent e) {
        dispose();
    }
});
```

```
Label firstName = new Label("First Name");
firstName.setBounds(20, 50, 80, 20);
```

```
Label lastName = new Label("Last Name");
lastName.setBounds(20, 80, 80, 20);
```

```
Label dob = new Label("Date of Birth");
dob.setBounds(20, 110, 80, 20);
```

```
firstNameTF = new TextField();
firstNameTF.setBounds(120, 50, 100, 20);
```

```
lastNameTF = new TextField();
lastNameTF.setBounds(120, 80, 100, 20);
```

```
dobTF = new TextField();
dobTF.setBounds(120, 110, 100, 20);
```

```
Button sbmt = new Button("Submit");
sbmt.setBounds(20, 160, 80, 30);
sbmt.addActionListener(
    new ActionListener() {
```

```

        public void actionPerformed(ActionEvent e) {
            String fnString = firstNameTF.getText();
            String lnString = lastNameTF.getText();
            String dobString = dobTF.getText();
            GP3_AwtForm awt2 = new GP3_AwtForm(fnString, lnString,
dobString);
        }
    }
};

```

```

Button reset = new Button("Reset");
reset.setBounds(140,160,80,30);
reset.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            firstNameTF.setText(null);
            lastNameTF.setText(null);
            dobTF.setText(null);
        }
    }
);

```

```

add(firstName);
add(lastName);
add(dob);
add(firstNameTF);
add(lastNameTF);
add(dobTF);
add(sbmt);
add(reset);

```

```

}

```

```
GP3_AwtForm(String fnString, String lnString, String dobString){
```

```
    setSize(300,300);
```

```
    setLayout(null);
```

```
    setVisible(true);
```

```
    setTitle("AWT Form Display");
```

```
    addWindowListener (new WindowAdapter() {
```

```
        public void windowClosing (WindowEvent e) {
```

```
            dispose();
```

```
        }
```

```
    });
```

```
    Label displayFirst = new Label("First Name: ");
```

```
    displayFirst.setBounds(20, 50, 80, 20);
```

```
    Label displayFName = new Label(fnString);
```

```
    displayFName.setBounds(100, 50, 80, 20);
```

```
    Label displayLast = new Label("Last Name: ");
```

```
    displayLast.setBounds(20, 80, 80, 20);
```

```
    Label displayLName = new Label(lnString);
```

```
    displayLName.setBounds(100, 80, 80, 20);
```

```
    Label displayDob = new Label("Date of Birth: ");
```

```
    displayDob.setBounds(20, 110, 80, 20);
```

```
    Label displayDobS = new Label(dobString);
```

```
    displayDobS.setBounds(100, 110, 80, 20);
```

```
    add(displayFirst);
```

```

add(displayFName);
add(displayLast);
add(displayLName);
add(displayDob);
add(displayDobS);

```

```

}

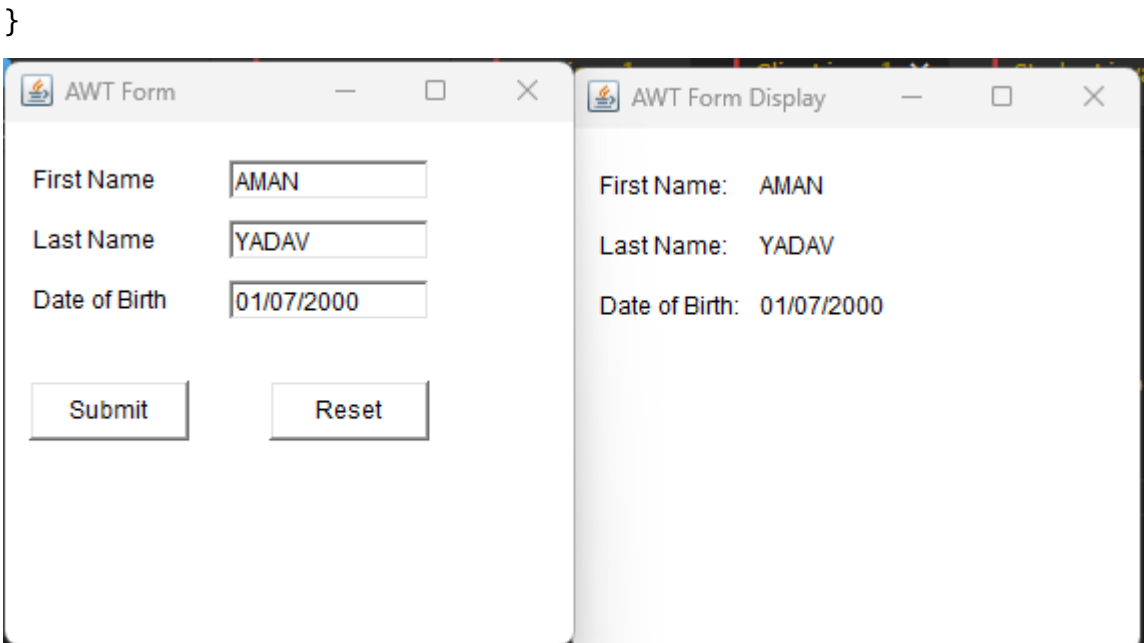
```

`@Override`

```

public void actionPerformed(ActionEvent evt) {}

```



**GP4:-** Create a class `MessageBox` that extends `Frame`. The class should have a constructor that takes a `String` as a parameter to construct a dialog box that displays the message and **OK & CANCEL** buttons. The dialog box should get closed when the cancel buttons is clicked. Provide some mechanism in the `MessageBox` class that can be used by the calling program to check which button was pressed by the user. The class should have functions to:

- Check which button was pressed by the user.
- Retrieve the string entered by the user, if user pressed **OK**, null if user pressed **CANCEL**.

Test this class to get a message from a user and display it on a `Frame` window.

**SOLN:-**

```

import java.awt.*;

```

```
import java.awt.event.*;

public class GP4_messageBox {
    public static void main(String args[]) {
        new GP4_messageBox("Hello");
        if(c == 1){

        }
    }
    private static Dialog d;
    private static int c;
    GP4_messageBox(String str) {

        Frame f= new Frame();
        f.setVisible(false);
        d = new Dialog(f , "Message Box", false);
        d.setLayout( new FlowLayout() );
        d.setSize(400,150);
        d.setVisible(true);
        Button ok = new Button ("OK");
        Button cancel = new Button ("Cancel");
        TextField message = new TextField();
        message.setText(str);

        d.add(message);
        d.add(ok);
        d.add(cancel);

        ok.addActionListener (
            new ActionListener() {
                public void actionPerformed((ActionEvent e) ) {
                    Frame s = new Frame();
                    s.setSize(300,300);
```

```

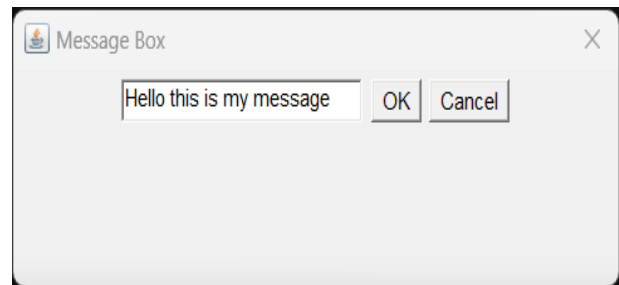
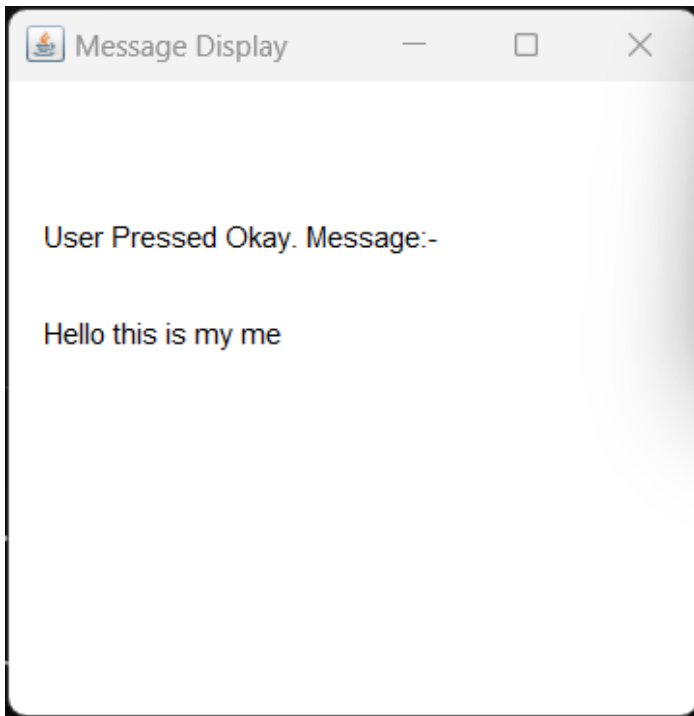
        s.setLayout(null);
        s.setVisible(true);
        s.setTitle("Message Display");
        Label lbl = new Label("User Pressed Okay. Message:-");
        lbl.setBounds(20, 80, 500, 30);
        s.add(lbl);
        Label display = new Label(message.getText());
        display.setBounds(20, 120, 100, 30);
        s.add(display);
        s.addWindowListener (new WindowAdapter() {
            public void windowClosing (WindowEvent e) {
                s.dispose();
            }
        });
    }
}

);

cancel.addActionListener (
    new ActionListener() {
        public void actionPerformed((ActionEvent e) {
            System.out.println("User Pressed CANCEL Button");
            System.out.println("GUI Closed");
            d.dispose();
            f.dispose();
        }
    }
);

```





// GP5:- Use AWT /Swings to develop a Contact Manager. The GUI application maintains a simple list of contacts storing contact name and number. It provides basic features of CRUD (Create, Update, Retrieve and Delete) as per user choice. The application also supports search feature.

// SOLN:-

//(Only Front End)

```
import javax.swing.*;
```

```
import java.awt.event.*;
```

```
import java.util.ArrayList;
```

```
public class GP5_contact {
```

```
    public static void main(String[] args) {
```

```
        new GP5_contact();
```

```
    }
```

```
    ArrayList<Contact> ContactList = new ArrayList<>();
```

```
    public GP5_contact(){
```

```
        JFrame f=new JFrame();
```

```

f.setSize(350,350);
f.setLayout(null);
f.setVisible(true);
f.setTitle("Contact Manager");
f.addWindowListener (new WindowAdapter() {
    public void windowClosing (WindowEvent e) {
        f.dispose();
    }
});

JLabel lbl = new JLabel("Contact Manager App");
lbl.setBounds(100, 60, 150, 30);
JButton c = new JButton("Create");
c.setBounds(60, 120, 100, 30);
JButton r = new JButton("Retrieve");
r.setBounds(170, 120, 100, 30);
JButton u = new JButton("Update");
u.setBounds(60, 160, 100, 30);
JButton d = new JButton("Delete");
d.setBounds(170, 160, 100, 30);
JButton all = new JButton("Display all Contacts");
all.setBounds(65, 200, 200, 30);
f.add(lbl);    f.add(c);  f.add(r);    f.add(u);    f.add(d);    f.add(all);

c.addActionListener (
    new ActionListener() {
        public void actionPerformed((ActionEvent e) {
            JFrame f=new JFrame();
            f.setSize(300,300);
            f.setLayout(null);
            f.setVisible(true);
            f.setTitle("Create");
            f.addWindowListener (new WindowAdapter() {

```

```

        public void windowClosing (WindowEvent e) {
            f.dispose();
        }
    });

    JLabel name = new JLabel("Name: ");
    name.setBounds(20, 70, 70, 30);
    JLabel mobile = new JLabel("Mobile No: ");
    mobile.setBounds(20, 110, 70, 30);
    JTextField nameTF = new JTextField();
    nameTF.setBounds(100, 70, 100, 30);
    JTextField mobileTF = new JTextField();
    mobileTF.setBounds(100, 110, 100, 30);
    JButton sbmt = new JButton("Submit");
    sbmt.setBounds(70, 150, 120, 30);
    f.add(name); f.add(mobile); f.add(nameTF); f.add(mobileTF);
f.add(sbmt);

    sbmt.addActionListener (
        new ActionListener() {
            public void actionPerformed( ActionEvent e ) {
                ContactList.add(new Contact(nameTF.getText(),
mobileTF.getText()));
                f.dispose();
            }
        }
    );
}

r.addActionListener (
    new ActionListener() {
        public void actionPerformed( ActionEvent e ) {

```

```

        JFrame f=new JFrame();
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
        f.setTitle("Retrieve");
        f.addWindowListener (new WindowAdapter() {
            public void windowClosing (WindowEvent e) {
                f.dispose();
            }
        });
    }
}
);

```

```

u.addActionListener (
    new ActionListener() {
        public void actionPerformed( ActionEvent e ) {
            JFrame f=new JFrame();
            f.setSize(300,300);
            f.setLayout(null);
            f.setVisible(true);
            f.setTitle("Update");
            f.addWindowListener (new WindowAdapter() {
                public void windowClosing (WindowEvent e) {
                    f.dispose();
                }
            });
        }
    }
);

```

```

d.addActionListener (
    new ActionListener() {

```

```

        public void actionPerformed((ActionEvent e) {
            JFrame f=new JFrame();
            f.setSize(300,300);
            f.setLayout(null);
            f.setVisible(true);
            f.setTitle("Delete");
            f.addWindowListener (new WindowAdapter() {
                public void windowClosing (WindowEvent e) {
                    f.dispose();
                }
            });
            JLabel lbl = new JLabel("Enter the Contact name to delete: ");
            lbl.setBounds(20, 50, 150, 30);
            JTextField TF = new JTextField();
            TF.setBounds(40, 90, 70, 30);
            JButton dl = new JButton("Delete Contact");
            dl.setBounds(60, 130, 80, 30);
            f.add(lbl); f.add(TF); f.add(dl);
        }
    }
};

```

```

all.addActionListener (
    new ActionListener() {
        public void actionPerformed( ActionEvent e ) {
            JFrame f=new JFrame();
            f.setSize(300,300);
            f.setLayout(null);
            f.setVisible(true);
            f.setTitle("Contact List");
            f.addWindowListener (new WindowAdapter() {
                public void windowClosing (WindowEvent e) {
                    f.dispose();
                }
            });
        }
    }
);

```

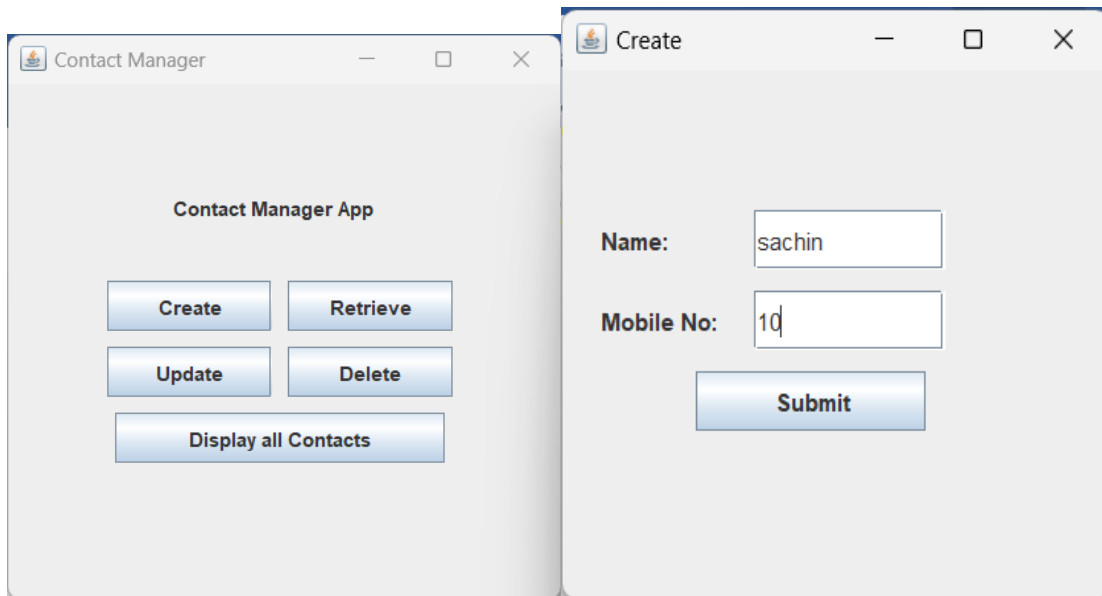
```

    }
    });
    for (int i=0; i<ContactList.size(); i++) {
        JLabel dl = new JLabel(ContactList.get(i).toString());
        dl.setBounds(20, 80+i*20, 100, 15);
        f.add(dl);
    }
}
} );
}

}

class Contact{
    private String Name, Mobile;
    Contact(String Name, String Mobile){
        this.Name = Name;
        this.Mobile = Mobile;
    }
    public String toString(){
        return Name+ " "+ Mobile;
    }
    public String getName(){
        return this.Name;
    }
}

```



// GP6:- Create a class MyTextEditor to simulate a notepad using Swings?

// SOLN:-

```
import javax.swing.*;
```

```
import java.awt.event.*;
```

```
public class GP6_textEditor extends JFrame implements ActionListener {
```

```
    public static void main(String args[]) {
```

```
        new GP6_textEditor();
```

```
    }
```

```
    JFrame f = new JFrame("Text Editor");
```

```
    JMenuBar m = new JMenuBar();
```

```
    JTextArea t = new JTextArea();
```

```
    GP6_textEditor() {
```

```
        JMenu m1 = new JMenu("File");
```

```
        JMenu m2 = new JMenu("Edit");
```

```
        m.add(m1);
```

```
        m.add(m2);
```

```
        JMenuItem m11 = new JMenuItem("new");
```

```

JMenuItem m12 = new JMenuItem("close");
JMenuItem m21 = new JMenuItem("cut");
JMenuItem m22 = new JMenuItem("copy");
JMenuItem m23 = new JMenuItem("paste");

m1.add(m11);
m1.add(m12);
m2.add(m21);
m2.add(m22);
m2.add(m23);

m11.addActionListener(this);
m12.addActionListener(this);
m21.addActionListener(this);
m22.addActionListener(this);
m23.addActionListener(this);

f.setSize(500, 500);
f.setVisible(true);
f.setJMenuBar(m);
f.add(t);
f.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        f.dispose();
    }
});
}

public void actionPerformed(ActionEvent e) {

    String s = e.getActionCommand();

    if (s.equals("new")) {

```



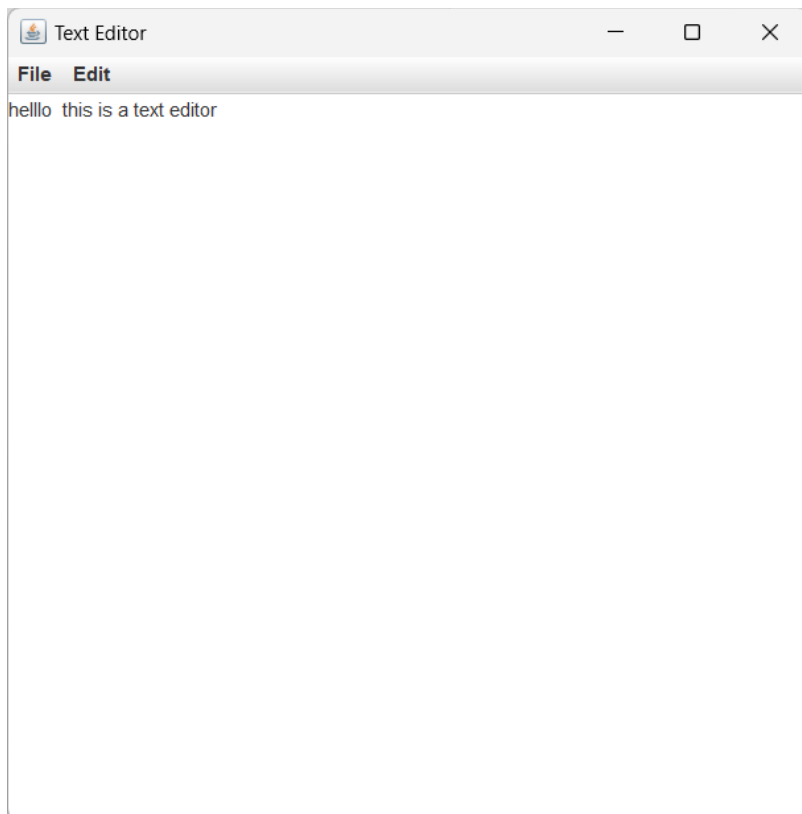
```

        t.setText("");
    }

    else if (s.equals("close")) {
        f.dispose();
    }

    else if (s.equals("cut")) {
        t.cut();
    } else if (s.equals("copy")) {
        t.copy();
    } else if (s.equals("paste")) {
        t.paste();
    }
}
}

```



HP1

A String tokenizer application to store the input string contents in a file. Read the file and count vowels, consonants and spaces in each line. Create another file to write the vowel and consonant

count besides each line. For eg:- Hi this is java(vowels-5, consonents-7, spaces- 3). I like studying it(vowels-6, consonants-9, spaces-4). Perform this operation using:

a) BufferedReader and BufferedWriter

b) FileReader and FileWriter

```
import java.io.*;

public class HP1 {
    public static void main(String[] args) {
        try {
            BufferedReader in = new BufferedReader(new FileReader("input.txt"));
            BufferedWriter out = new BufferedWriter(new FileWriter("output.txt"));




            // Same can be down using FileReader and FileWriter like

            // FileReader in = new FileReader("input.txt");
            // FileWriter out = new FileWriter("output.txt");
            String line;
            while ((line = in.readLine()) != null) {
                int vowelCount = 0;
                int consonantCount = 0;
                int spaceCount = 0;
                for (int i = 0; i < line.length(); i++) {
                    char ch = line.charAt(i);
                    if (ch == 'a' || ch == 'e' || ch == 'i' ||
                        ch == 'o' || ch == 'u' || ch == 'A' || ch == 'E'
                        || ch == 'I' || ch == 'O' || ch == 'U') {
                        vowelCount++;
                    } else if (ch == ' ') {
                        spaceCount++;
                    } else if (ch >= 'a' && ch <= 'z' || ch >= 'A' && ch <= 'Z') {
                        consonantCount++;
                    }
                }
            }
        }
    }
}
```

```

        out.write(line + " (vowels: " + vowelCount +
            ", consonants: " + consonantCount + ", spaces: "
            + spaceCount + ")");
        out.newLine();
    }
    in.close();
    out.close();
} catch (IOException e) {
    e.printStackTrace();
}
}

```

 HP1.java	25-01-2023 11:50	Java Source File	2 KB
 input.txt	25-01-2023 12:08	Text Document	1 KB
 output.txt	25-01-2023 12:09	Text Document	1 KB

## HP2

A File Parser a file and store the following text in it-

“Dwelling and speedily ignorant any steepest. Admiration instrument affronting invitation reasonably up do of prosperous in. Shy saw declared age debating ecstatic man. Call in so want pure rank am dear were. Remarkably to continuing in surrounded diminution on. In unfeeling existence objection immediate repulsive on he in. Imprudence comparison uncommonly me he difficulty diminution resolution. Likewise proposal differed scarcely dwelling as on raillery. September few dependent extremity own continued and ten prevailed attending. Early to weeks we could.

Unpleasant astonished an diminution up partiality. Noisy an their of meant. Death means up civil do an offering wound of. Called square an in afraid direct. Resolution diminution conviction so mr at unpleasing simplicity no. No it as breakfast up conveying earnestly immediate principle. Him son disposed produced humored overcame she bachelor improved. Studied however out wishing but inhabit fortune windows.”

Accept a SearchToken from the user. Open the file and read it using RandomFileAccess and search and display total occurrences of the search string in given text.

```

import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Scanner;

```

```

public class HP2 {

    public static void main(String[] args) {

        // Accept a search token from the user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a search token: ");
        String searchToken = scanner.nextLine();

        try {

            // Open the file using RandomAccessFile
            RandomAccessFile file = new RandomAccessFile("input.txt", "r");

            // Read the file and search for the search token
            String line;
            int count = 0;
            while ((line = file.readLine()) != null) {
                int index = line.indexOf(searchToken);
                while (index != -1) {
                    count++;
                    index = line.indexOf(searchToken, index + 1);
                }
            }

            // Display the total number of occurrences of the search token
            System.out.println("Total occurrences: " + count);

            // Close the file
            file.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```
PS C:\Users\Aman Yadav> java HP2.java
Enter a search token: java
Total occurrences: 1
PS C:\Users\Aman Yadav> 
```

### HP3

Define your Student Class to Serialize objects of student class into separate files and byte streams using Serializable interface. Make use of the serialVersionUID field and declare few variables as transient. Deserialize the objects and store them into an array of objects on another JVM instance and perform sorting based on parameter of user's choice using Comparator.

Note: You will have to make separate comparator classes for different data member based sorting.

```
import java.io.*;
```

```
public class HP3 {
    public static void main(String[] args) {
        try {
            Student[] students = {
                new Student("aman", 20, 89.5),
                new Student("Babu", 22, 73.0),
                new Student("Chintu", 18, 95.0)
            };

            FileOutputStream fileOut = new FileOutputStream("students.ser");
            ObjectOutputStream out = new ObjectOutputStream(fileOut);

            for (Student student : students) {
                out.writeObject(student);
            }

            out.close();
            fileOut.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

try {
    FileInputStream fileIn = new FileInputStream("students.ser");
    ObjectInputStream in = new ObjectInputStream(fileIn);

    Student student;
    while ((student = (Student) in.readObject()) != null) {
        System.out.println(student);
    }
    in.close();
    fileIn.close();
} catch (EOFException e) {
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
}
}
}

```

```

class Student implements Serializable {
    // Declare a serialVersionUID field
    private static final long serialVersionUID = 1L;
    private String name;
    private int age;
    private transient double grade;

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

    public String toString() {
        return "name: " + name + ", " +
            "age: " + age + ", " +

```

```

        "grade: " + grade;
    }
}

```

```

PS C:\Users\Aman Yadav> java HP3
name: aman, age: 20, grade: 0.0
name: Babu, age: 22, grade: 0.0
name: Chintu, age: 18, grade: 0.0

```

// HP4)A Java application that uses JDBC to connect to a database containing table that holds data for students of your class.The application should allow the user to:a)Add a record b)Search for a record c)Modify an existing record Make use of Prepared Statement.Extend the application to call a backend procedure/function.The procedure should take ID field as input parameters and return details of related record.

```

import java.sql.*;
import java.util.*;

public class IP3 {
    public static void addInDb() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the Student Details: Id, Name and Course ");
        int id = sc.nextInt();
        String name = sc.next();
        String course = sc.next();
        try {
            Class.forName("org.sqlite.JDBC");
            Connection con =
DriverManager.getConnection("jdbc:sqlite:C://sqlite//myclg.db");
            PreparedStatement pstmt = con.prepareStatement("insert into students
values(?,?,?)");
            pstmt.setInt(1, id);
            pstmt.setString(2, name);
            pstmt.setString(3, course);
            pstmt.executeUpdate();
            con.close();
            System.out.println("!!! Record Saved !!!");
        } catch (Exception e) {

```

```
        System.out.println(e);
        System.exit(0);
    }
}
```

```
public static void deleteInDb() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Student id to delete");
    int id = sc.nextInt();
    try {
        Class.forName("org.sqlite.JDBC");
        Connection con =
DriverManager.getConnection("jdbc:sqlite:C://sqlite//myclg.db");
        PreparedStatement pstmt = con.prepareStatement("delete from students
where id=?");
        pstmt.setInt(1, id);
        pstmt.executeUpdate();
        con.close();
        System.out.println("!!! Record Deleted !!!");
    } catch (Exception e) {
        System.out.println(e);
        System.exit(0);
    }
}
```

```
public static void updateInDb() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Student id to to update its Data:");
    int id = sc.nextInt();
    try {
        Class.forName("org.sqlite.JDBC");
        Connection con =
DriverManager.getConnection("jdbc:sqlite:C://sqlite//myclg.db");
        System.out.println("Enter the choice: ");
```



```

        System.out.println("1) Update Name -");
        System.out.println("2) Update Course -");
        int choice = sc.nextInt();
        switch (choice) {
            case 1:
                String nm = sc.next();
                PreparedStatement pstm = con.prepareStatement("update students
setname=? where id=?");
                pstm.setInt(2, id);
                pstm.setString(1, nm);
                pstm.executeUpdate();
                con.close();
                System.out.println("!!! RECORD UPDATED (Name) !!!");
                break;
            case 2:
                String crs = sc.next();
                PreparedStatement pstm2 = con.prepareStatement("update
students setcourse=? where id=?");
                pstm2.setInt(2, id);
                pstm2.setString(1, crs);
                pstm2.executeUpdate();
                con.close();
                System.out.println("!!! RECORD UPDATED (course) !!!");
                break;
            default:
                System.out.println("Enter Valid Choice!!!");

                break;
        }
    } catch (Exception e) {
        System.out.println(e);
        System.exit(0);
    }
}

```

```

public static void searchInDb() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Student id to get the the Student Detail:");
    int id = sc.nextInt();
    try {
        Class.forName("org.sqlite.JDBC");
        Connection con =
DriverManager.getConnection("jdbc:sqlite:C://sqlite//myclg.db");
        PreparedStatement pstmt = con.prepareStatement("select * from students
whereid=?");
        pstmt.setInt(1, id);
        ResultSet rs = pstmt.executeQuery();
        System.out.println("Id: " + rs.getInt(1) + " Name: " + rs.getString(2)
+ " Course: " + rs.getString(3));
        con.close();
    } catch (Exception e) {
        System.out.println("Invalid Input no such Student found" + e);
    }
}

```

```

public static void main(String[] args) throws Exception {
    Scanner sc = new Scanner(System.in);
    do {
        System.out.println("Select your choice: ");
        System.out.println("1) ----Add new Student---");
        System.out.println("2) ----Delete Student ---");
        System.out.println("3) ----Update Existing Student---");
        System.out.println("4) ----Search for Existing Student and Display its
Details---");
        System.out.println("5) ---- Exit---");
        int x = sc.nextInt();
        switch (x) {

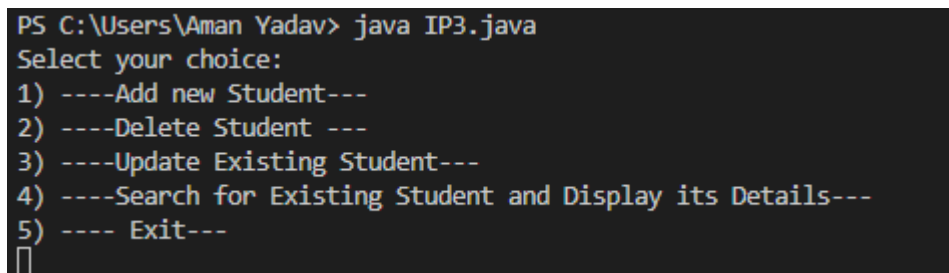
            case 1:

```

```

        addInDb();
        break;
    case 2:
        deleteInDb();
        break;
    case 3:
        updateInDb();
        break;
    case 4:
        searchInDb();
        break;
    case 5:
        System.exit(0);
    default:
        System.out.println("Invalid input");
        break;
    }
} while (true);
}
}

```



```

PS C:\Users\Aman Yadav> java IP3.java
Select your choice:
1) ----Add new Student---
2) ----Delete Student ---
3) ----Update Existing Student---
4) ----Search for Existing Student and Display its Details---
5) ---- Exit---

```

HP5) Create an RMI application that exposes a remote object School. It should have two remote methods admit and search. The method admit should add a student's record in the list of available students and search should return the details of student on basis of roll number entered or raise an exception in case of invalid roll number. Demonstrate the use of these methods in a RMI client application

//student.java

```

import java.rmi.Remote;
import java.rmi.RemoteException;

```

```
interface SchoolInterface extends Remote {  
    public void admit(String rollNo, String name) throws RemoteException;  
  
    public Student search(String rollNo) throws RemoteException;  
}
```

```
class Student implements java.io.Serializable {  
    private static final long serialVersionUID = 1L;  
    private String rollNo;  
    private String name;  
  
    public Student(String rollNo, String name) {  
        this.rollNo = rollNo;  
        this.name = name;  
    }  
  
    public String getRollNo() {  
        return rollNo;  
    }  
  
    public String getName() {  
        return name;  
    }  
}
```

//Server.java

```
import java.rmi.*;  
import java.rmi.registry.LocateRegistry;  
import java.rmi.registry.Registry;  
import java.rmi.server.*;  
import java.util.ArrayList;  
import java.util.List;
```

```
public class Server {  
    public static void main(String[] args) {  
        try {  
            School school = new School();  
            Registry registry = LocateRegistry.createRegistry(5000);  
            registry.rebind("SchoolService", school);  
  
            System.out.println("School Server is ready.");  
        } catch (Exception e) {  
            System.out.println("School Server failed: " + e);  
        }  
    }  
}
```

```
class School extends UnicastRemoteObject implements SchoolInterface {  
    private static final long serialVersionUID = 1L;  
    private List<Student> students;  
  
    public School() throws RemoteException {  
        students = new ArrayList<Student>();  
    }  
  
    public void admit(String rollNo, String name) throws RemoteException {  
        Student student = new Student(rollNo, name);  
        students.add(student);  
    }  
  
    public Student search(String rollNo) throws RemoteException {  
        for (Student student : students) {  
            if (student.getRollNo().equals(rollNo)) {  
                return student;  
            }  
        }  
    }  
}
```

```

    }
    throw new RemoteException("Invalid roll number");
}
}

```

//Client.java

```

import java.rmi.Naming;

public class Client {
    public static void main(String[] args) {
        try {
            SchoolInterface school = (SchoolInterface)
Naming.lookup("rmi://localhost:5000/SchoolService");
            school.admit("1", "sourav Gusain");
            school.admit("2", "Bablu Singh");
            Student student = school.search("1");
            System.out.println("Name: " + student.getName());
        } catch (Exception e) {
            System.out.println("School Client failed: " + e);
        }
    }
}

```

```

PS C:\Users\Aman Yadav> java Server
School Server is ready.

```

```

PS C:\Users\Aman Yadav> java Client
Name: AMAN YADAV
PS C:\Users\Aman Yadav>

```