**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No.01**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE : Complex Arithmetic** |

**AIM:** Write a program to perform addition, subtraction, multiplication and division of two complex numbers. Demonstrate the use of classes and objects.

Variations :

Implementation of Program with One class

Accessibility with static and non-static methods within class and outside class.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. E. Balagurusamy , “Programming with Java” McGraw-Hill.
2. Sachin Malhotra, Saurabh Choudhary, “Programming in Java”, Oxford Publications.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

The Scanner class is a class in java.util, which allows the user to read values of various types. There are far more methods in class Scanner than you will need in this course. We only cover a small useful subset, ones that allow us to read in numeric values from either the keyboard or file without having to convert them from strings and determine if there are more values to be read.

Scanner in = new Scanner(System.in);  // System.in is an InputStream

 Numeric and String Methods

|  |  |
| --- | --- |
| **Method** | **Returns** |
| int nextInt() | Returns the next token as an int. If the next token is not an integer,InputMismatchException is thrown. |
| long nextLong() | Returns the next token as a long. If the next token is not an integer,InputMismatchException is thrown. |
| float nextFloat() | Returns the next token as a float. If the next token is not a float or is out of range, InputMismatchException is thrown. |
| double nextDouble() | Returns the next token as a long. If the next token is not a float or is out of range, InputMismatchException is thrown. |
| String next() | Finds and returns the next complete token from this scanner and returns it as a string; a token is usually ended by whitespace such as a blank or line break. If not token exists,NoSuchElementException is thrown. |
| String nextLine() | Returns the rest of the current line, excluding any line separator at the end. |
| void close() | Closes the scanner. |

The Scanner looks for tokens in the input. A token is a series of characters that ends with what Java calls whitespace. A whitespace character can be a blank, a tab character, a carriage return. Thus, if we read a line that has a series of numbers separated by blanks, the scanner will take each number as a separate token. .

The numeric values may all be on one line with blanks between each value or may be on separate lines.   Whitespace characters (blanks or carriage returns) act as separators.  The next method returns the next input value as a string, regardless of what is keyed.  For example, given the following code segment and data

* int number = in.nextInt();
* float real = in.nextFloat();
* long number2 = in.nextLong();
* double real2 = in.nextDouble();
* String string = in.next();

**Class Diagram:**

**Same Class:**

|  |  |
| --- | --- |
| Class name | exp1a |
| Variables | - |
| Functions | add (), sub(), multi(), div(), main() |

**Different Class:**

|  |  |
| --- | --- |
| Class name | exp1 |
| Variables | - |
| Functions | main() |

|  |  |
| --- | --- |
| Class name | complex |
| Variables | r : double  i : double |
| Functions | add (), sub(), multi(), div(), main() |

**Algorithm:**

1. Start
2. Get input for num1
3. Get input for num2
4. Print addition : call add()
5. Print subtraction : call sub()
6. Print multiplication: call multi()
7. Print Division : call div()

Add()

1. Start
2. Add real numbers together
3. Add imaginary numbers together
4. Return the respective complex number

Sub()

1. Start
2. Subtract real numbers together
3. Subtract imaginary numbers together
4. Return the respective complex number

Multi()

1. Start
2. Multiply the given complex numbers using the mathematical formula for multiplication of complex numbers.
3. Return the respective complex number

Div()

1. Start
2. Divide the given complex numbers using the mathematical formula for Division of complex numbers.
3. Return the respective complex number

**Implementation details:**

**Same Class:**

import java.util.Scanner;

class exp1a {

*static* void add(double *r1*,double *i1*,double *r2*,double *i2*) {

System.out.println("Addition Result:");

System.out.println("The Real part is: " + (*r1*+*r2*) + " Imaginary part is: " + (*i1* + *i2*));

}

*static* void sub(double *r1*,double *i1*,double *r2*,double *i2*) {

System.out.println("Subtraction Result:");

System.out.println("The Real part is: " + (*r1* - *r2*) + " Imaginary part is: " + (*i1* - *i2*));

}

*static* void multi(double *r1*,double *i1*,double *r2*,double *i2*) {

System.out.println("Multiplication Result:");

double real, img;

real = (*r1* \* *r2*) - (*i1* \* *i2*);

img = (*r1* \* *i2*) + (*i1* \* *r2*);

System.out.println("The Real part is: " + real + " Imaginary part is: " + img);

}

*static* void div(double *r1*,double *i1*,double *r2*,double *i2*) {

System.out.println("Division Result:");

double den = (*r2* \* *r2*) + (*i2* \* *i2*);

double real = (*r1* \* *r2*) + (*i1* \* *i2*);

double img = (*i1* \* *r2*) - (*r1* \* *i2*);

System.out.println("The Real part is: " + real / den + " Imaginary part is: " + img / den);

}

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

Scanner sc = *new* Scanner(System.in);

System.out.print("Enter Real Part for num1: ");

double real1 = sc.nextDouble();

System.out.print("Enter Img Part for num1: ");

double img1 = sc.nextDouble();

System.out.print("Enter Real Part for num2: ");

double real2 = sc.nextInt();

System.out.print("Enter Img Part for num2: ");

double img2 = sc.nextInt();

exp1a.add(real1, img1, real2, img2);

exp1a.sub(real1, img1, real2, img2);

*// Calling non-static methods*

exp1a.multi(real1, img1, real2, img2);

exp1a.div(real1, img1, real2, img2);

sc.close();

}

}

**Different Class:**

import java.util.Scanner;

class exp1 {

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

Scanner sc = *new* Scanner(System.in);

System.out.print("Enter Real Part for num1: ");

double real = sc.nextDouble();

System.out.print("Enter Img Part for num1: ");

double img = sc.nextDouble();

*// Creating a complex number num1*

complex num1 = *new* complex(real, img);

System.out.print("Enter Real Part for num2: ");

real = sc.nextInt();

System.out.print("Enter Img Part for num2: ");

img = sc.nextInt();

*// Creating a complex number num2*

complex num2 = *new* complex(real, img);

*// Calling static methods*

complex.add(num1, num2);

complex.sub(num1, num2);

*// Calling non-static methods*

num1.multi(num1, num2);

num2.div(num1, num2);

sc.close();

}

}

class complex {

double r, i;

complex(double *r*, double *i*) {

this.r = *r*;

this.i = *i*;

}

*static* void add(*complex* *num1*, *complex* *num2*) {

System.out.println("Addition Result:");

System.out.println("The Real part is: " + (*num1*.r + *num2*.r) + " Imaginary part is: " + (*num1*.i + *num2*.i));

}

*static* void sub(*complex* *num1*, *complex* *num2*) {

System.out.println("Subtraction Result:");

System.out.println("The Real part is: " + (*num1*.r - *num2*.r) + " Imaginary part is: " + (*num1*.i - *num2*.i));

}

void multi(*complex* *num1*, *complex* *num2*) {

System.out.println("Multiplication Result:");

double real, img;

real = (*num1*.r \* *num2*.r) - (*num1*.i \* *num2*.i);

img = (*num1*.r \* *num2*.i) + (*num1*.i \* *num2*.r);

System.out.println("The Real part is: " + real + " Imaginary part is: " + img);

}

void div(*complex* *num1*, *complex* *num2*) {

System.out.println("Division Result:");

double den = (*num2*.r \* *num2*.r) + (*num2*.i \* *num2*.i);

double real = (*num1*.r \* *num2*.r) + (*num1*.i \* *num2*.i);

double img = (*num1*.i \* *num2*.r) - (*num1*.r \* *num2*.i);

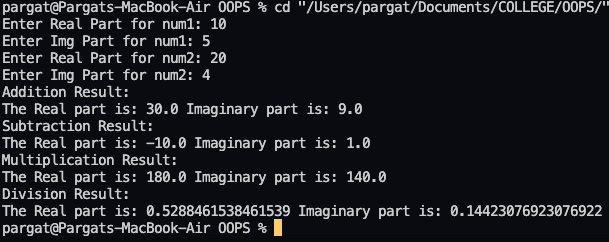
System.out.println("The Real part is: " + real / den + " Imaginary part is: " + img / den);

}

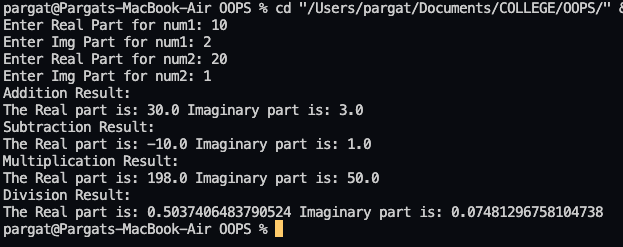
}

**Output:**

**Same Class:**

****

**Different Class:**



**Conclusion:**

In this experiment, we learnt how to implement program with one class and multiple classes. Along with accessibility with static and non-static methods within class and outside class.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions:**

**Q.1 Write a program to find the area and circumference of a circle using two classes.**

Code:

import java.util.\*;

class post1{

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

Scanner sc=*new* Scanner(System.in);

System.out.print(">>Enter Cirle Radius: ");

double r=sc.nextDouble();

circle.area(r);

circle.circum(r);

}

}

class circle{

*static* void area(double *r*){

System.out.println(">>The Area of circle is: "+(3.14\**r*\**r*));

}

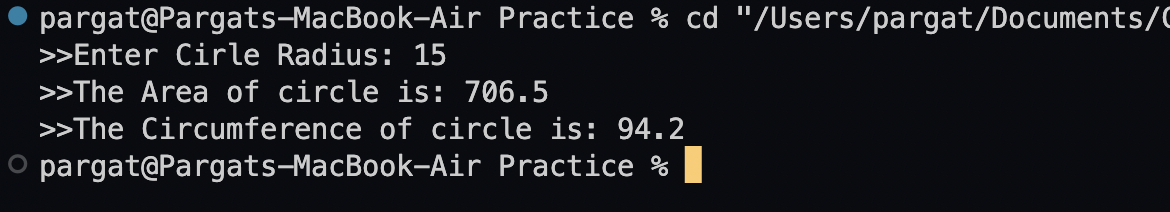
*static* void circum(double *r*){

System.out.println(">>The Circumference of circle is: "+(2\*3.14\**r*));

}

}

Output:



**Q.2 Write the output of following program**

1. **public** **class** BreakExample2 {
2. **public** **static** **void** main(String[] args) {
3. //outer loop
4. **for**(**int** i=1;i<=3;i++){
5. //inner loop
6. **for**(**int** j=1;j<=3;j++){
7. **if**(i==2&&j==2){
8. //using break statement inside the inner loop
9. **break**;
10. }
11. System.out.println(i+" "+j);
12. }
13. }
14. }
15. }

**Output:**

1 1

1 2

1 3

2 1

3 1

3 2

3 3

**Q.3 Why is Java known as a platform independent language?**

Java is based on Write-Once-Run-Anywhere concept that makes it Platform independent. It is platform independent because the program written in it is not directly converted into machine code but instead is converted into byte code by Java compiler, this byte code is then converted into machine readable code by Java Virtual Machine (JVM). JDK including JVM must be installed in a platform.

Q.4 Write a recursive static method for calculation of gcd of a number.

class post2 {

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

int a = 15, b = 150;

System.out.println("G.C.D is: "+gcd(a,b));

}

*public* *static* int gcd(int *a*, int *b*) {

*if* (*b* != 0)

*return* gcd(*b*, *a* % *b*);

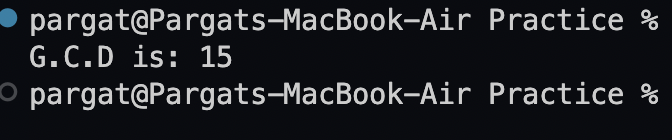
*else*

*return* *a*;

}

}

Output:



**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No. 02**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE : Control Statement** |

**AIM:** Create a class myMath. The class contains the following static methods.

i) power (x, y) – to compute x y

ii) fact (x) – to compute x!

Write a program to find the following series.

* ex = 1 +(x/1!)+ (x2/2!) + (x3/3!) + (x4/4!) + … upto n terms (n given by user).
* (1+x)n = 1 +(nx/1!)+ ((n(n-1)x2)/2!) ........ upto n terms (n given by user).

**(Do not make use of inbuilt functions. Use the functions of user defined class MyMath.)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. E. Balagurusamy , “Programming with Java” McGraw-Hill.
2. Sachin Malhotra, Saurabh Choudhary, “Programming in Java”, Oxford Publications.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts**

Java basic constructs (like if else statement, control structures, and data types

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −

|  |  |
| --- | --- |
| **Sr.No.** | **Loop & Description** |
| 1 | [**while loop**](https://www.tutorialspoint.com/java/java_while_loop.htm)  Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body. |
| 2 | [**for loop**](https://www.tutorialspoint.com/java/java_for_loop.htm)  Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| 3 | [**do...while loop**](https://www.tutorialspoint.com/java/java_do_while_loop.htm)  Like a while statement, except that it tests the condition at the end of the loop body. |

**Loop Control Statements**

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Java supports the following control statements. Click the following links to check their details.

|  |  |
| --- | --- |
| **Sr.No.** | **Control Statement & Description** |
| 1 | [**break statement**](https://www.tutorialspoint.com/java/java_break_statement.htm)  Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch. |
| 2 | [**continue statement**](https://www.tutorialspoint.com/java/java_continue_statement.htm)  Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. |

**Class Diagram:**

|  |  |
| --- | --- |
| Class name | exp2 |
| Variables | - |
| Functions | main() |

|  |  |
| --- | --- |
| Class name | myMath |
| Variables | - |
| Functions | power(), fact(), expo(), series() |

**Algorithm:**

1. Start
2. Print 2^5 power : call power()
3. Print 5 factorial : call fact()
4. Take user input for n and x
5. Print result (1+x)^n : call series()
6. Print e^x result : call expo()

power()

1. Start
2. x and n are parameters
3. Initialize ans = 1
4. Loop n times
5. Ans = ans \* x
6. return ans

fact()

1. Start
2. n is a parameter
3. if n>1

* return n \* fact(n-1)

1. else return 1.

series()

1. Start
2. x and n are parameters
3. initialize ans = 0
4. loop n+1 times

* ans = ans + (power(x, i) \* (fact(n) / fact(n - i))) / fact(i)

1. return ans

expo()

1. Start
2. x and n are parameters
3. initialize ans = 0
4. loop n times

* ans = ans + (power(x, i) / fact(i))

1. return ans

**Implementation details:**

import java.util.Scanner;

class exp2 {

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

Scanner sc = *new* Scanner(System.in);

System.out.println("2^5 is: " + myMath.power(2, 5));

System.out.println("5 factorial is: " + myMath.fact(5));

System.out.println("Enter n :");

int n = sc.nextInt();

System.out.println("Enter x: ");

double x = sc.nextDouble();

System.out.println("(1+x)^n : "+myMath.series(n, x));

System.out.println("e^x for first n terms: "+myMath.expo(n, x));

}

}

class myMath {

*static* double power(double *x*, int *y*) {

double ans = 1;

*for* (int i = 0; i < *y*; i++)

ans = ans \* *x*;

*return* ans;

}

*static* int fact(int *n*) {

*if* (*n* > 1)

*return* *n* \* fact(*n* - 1);

*else*

*return* 1;

}

*static* double expo(int *n*, double *x*) {

double ans = 0;

*for* (int i = 0; i < *n*; i++)

ans = ans + (power(*x*, i) / fact(i));

*return* ans;

}

*static* double series(int *n*, double *x*) {

double ans = 0;

*for* (int i = 0; i <= *n*; i++)

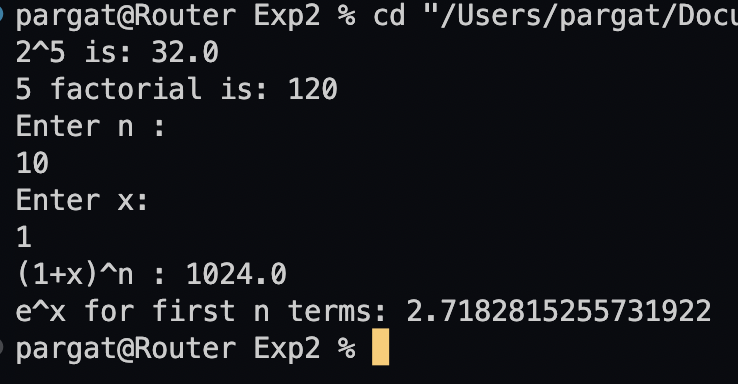
ans = ans + (power(*x*, i) \* (fact(*n*) / fact(*n* - i))) / fact(i);

*return* ans;

}

}

**Output:**



**Conclusion:**

Developed custom math functions in java and implement it using static class concept.

Successfully executed the given problem statement.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

Q.1 Write a program to find the largest of three numbers using the if-else construct.

Code:

import java.util.Scanner;

class exp2post {

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

Scanner sc=*new* Scanner(System.in);

System.out.println("Enter 3 numbers");

double a=sc.nextDouble();

double b=sc.nextDouble();

double c=sc.nextDouble();

*if*(a>b){

*if*(a>c)

System.out.println(a+" is the largest");

*else*

System.out.println(c+" is the largest");

}

*else*{

*if*(b>c)

System.out.println(b+" is the largest");

*else*

System.out.println(c+" is the largest");

}

}

}

Output:

Text

Description automatically generated

Q.2 Write a program to determine the sum of the following series for a given value of n:

1+½+⅓+....+1/n

Code:

import java.util.Scanner;

class ex2post2 {

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

Scanner sc=*new* Scanner(System.in);

System.out.print("Enter n: ");

int n=sc.nextInt();

double ans=0;

*for*(int i=1;i<=n;i++)

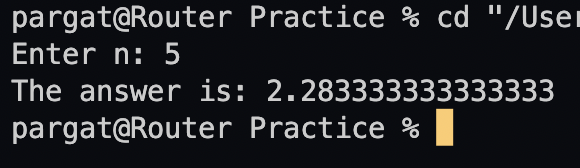
ans=ans + 1.0/i;

System.out.println("The answer is: "+ans);

}

}

Output:

****

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No. 03**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :Multi-dimensional Arrays (Jagged Array)** |

**AIM:** Write a program which stores information about n players in a two dimensional array. The array should contain the number of rows equal to the number of players. Each row will have a number of columns equal to the number of matches played by that player which may vary from player to player. The program should display player number (index +1), runs scored in all matches and its batting average as output. (It is expected to assign columns to each row dynamically after getting value from the user.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. E. Balagurusamy , “Programming with Java” McGraw-Hill.
2. Sachin Malhotra, Saurabh Choudhary, “Programming in Java”, Oxford Publications.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Arrays

**Multi-Dimensional Array**:

10 12 43 11 22

20 45 56 1 33

30 67 32 14 44

40 12 87 14 55

50 86 66 13 66

60 53 44 12 11

A multi-dimensional array is one that can hold all the values above. You set them up like this:

**int[ ][ ] numbers = new int[**6**][**5**];**

The first set of square brackets is for the rows and the second set of square brackets is for the columns. In the above line of code, we're telling Java to set up an array with 6 rows and 5 columns.

aryNumbers[0][0] = 10;  
aryNumbers[0][1] = 12;  
aryNumbers[0][2] = 43;  
aryNumbers[0][3] = 11;  
aryNumbers[0][4] = 22;

So the first row is row 0. The columns then go from 0 to 4, which is 5 items.

**Class Diagram:**

|  |  |
| --- | --- |
| Class name | Exp3 |
| Variables | - |
| Functions | main() |

**Algorithm:**

1. Start
2. Take n Input
3. Initialize 2D array row
4. Loop n times

* Take number of matches as input
* Initialize column as number of matches +1
* Store first element of every row as jersey number
* Loop m times
* Take number of runs as input
* Store in 2D array

1. Print Player Stats

* Print Player Number
* Print number of matches
* Calculate total runs and print
* Print Batting Average i.e Total runs/ Number of matches

**Implementation details:**

import java.util.Scanner;

class exp3 {

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

Scanner sc=*new* Scanner(System.in);

System.out.print("Enter the value of n: ");

int n=sc.nextInt();

int team[][]=*new* int[n][];

*for*(int i=0;i<n;i++){

System.out.print("\nEnter the number of matches for Player "+(i+1)+" : ");

int m=sc.nextInt();

team[i]=*new* int[m+1];

team[i][0]=i+1;

*for*(int j=0;j<m;j++){

System.out.print("Enter runs in match "+(j+1)+" : ");

team[i][j+1]=sc.nextInt();

}

}

System.out.println();

*for*(int i=0;i<n;i++){

System.out.println("Player "+(i+1)+" Stats");

System.out.println("------------------------\nNumber of Matches Played : "+(team[i].length-1));

double totalRuns=0;

*for*(int j=1;j<team[i].length;j++){

System.out.println("Runs in Match "+(j)+" : "+team[i][j]);

totalRuns+=team[i][j];

}

System.out.println("Total Runs : "+totalRuns);

System.out.println("Batting Average : "+totalRuns/(team[i].length-1));

System.out.println();

}

}

}

**Output:**

Text

Description automatically generated

**Conclusion:**

This experiment involved studying of multidimensional and jagged arrays. The latter was used for implementing the given problem statement and achieving desired results.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

**Q.1 Create a jagged array of integers. This array should consist of two 2-D arrays. First 2-D array should contain 3 rows having length of 4,3,and 2 respectively. Second 2-D array should contain 2 rows with length 3 and 4 respectively.**

class post3q1 {

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

int arr1[][]=*new* int[3][];

arr1[0]=*new* int[4];

arr1[1]=*new* int[3];

arr1[2]=*new* int[2];

int arr2[][]=*new* int[2][];

arr2[0]=*new* int[3];

arr2[1]=*new* int[4];

System.out.println("Array 1");

*for*(int i=0;i<arr1.length;i++){

*for*(int j=0;j<arr1[i].length;j++)

System.out.print(arr1[i][j]+" ");

System.out.println();

}

System.out.println("\nArray 2");

*for*(int i=0;i<arr2.length;i++){

*for*(int j=0;j<arr2[i].length;j++)

System.out.print(arr2[i][j]+" ");

System.out.println();

}

}

}

**Output:**

**Text

Description automatically generated**

**Q.2 Consider the following code**

int number[] = new int[5];

After execution of this statement, which of the following are true?

(A) number[0] is undefined

(B) number[5] is undefined

(C) number[4] is null

(D) number[2] is 0

(E) number.length() is 5

(i) (C) & (E)

(ii) (A) & (E)

(iii) (E)

(iv) (B), (D) & (E)

**Ans: (iv) (B), (D) & (E)**

**Q.3 Write a program to create an array where ith row has i columns.**

class post3q1 {

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

int arr1[][]=*new* int[5][];

*for*(int i=0;i<arr1.length;i++)

arr1[i]=*new* int[i];

*for*(int i=0;i<arr1.length;i++){

*for*(int j=0;j<arr1[i].length;j++)

System.out.print(arr1[i][j]+" ");

System.out.println();

}

}

}

**Graphical user interface, text

Description automatically generated**

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No. 04**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :An Array of Objects** |

**AIM:** Write a program which accepts information about n no of customers from user .Create an array of objects to store account\_id ,name,balance.

Your program should provide following functionalities

1. To add account
2. To delete any account detail
3. To display account details.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1:** Understand the features of object oriented programming compared with procedural approach with C++ and Java

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGraw-Hill.
2. Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

**Arrays of Objects:**

Unlike traditional array which store values like string, integer, boolean, etc. array of objects stores objects. The array elements store the location of reference variables of the object.

**For example:**

class Student {  
   int rno;

String name;

float avg;  
}

Student(int r, String name, float average)

{

rno=r;

this.name=name;

avg=average;

}

Student studentArray[] = new Student[n];

* The above statement creates the array which can hold references to n number of Student objects. It doesn't create the Student objects themselves. They have to be created separately using the constructor of the Student class. The studentArray contains n number of memory spaces in which the address of n Student objects may be stored.

for ( int i=0; i<studentArray.length; i++) {  
studentArray[i]=new Student(r,name,average);  
}

* The above for loop creates n Student objects and assigns their reference to the array elements. Now, a statement like the following would be valid.

studentArray[i].r=1001;

.

**Class Diagram:**

|  |  |
| --- | --- |
| Class name | Exp4 |
| Variables | - |
| Functions | main() |

|  |  |
| --- | --- |
| Class name | acccount |
| Variables | Int id, String name, double balance |
| Functions | - |

**Algorithm:**

1. Create a class Account with attributes int is, String name and float Balance.

2. Create a constructor for this class.

3. Create a public class Expt\_4.

4. Define the main method in this class.

5. Create an object of Scanner class.

6. Get the total number of customers from the user and declare the array of same length.

7. Define a while loop.

8. In this while loop get choice from the user to Add, Delete, Display the account or to Exit.

9. If user selects option 1.

10. Get the number of accounts to be added initially.

11. Get account number, account holder name and balance.

12. Store this in the array of objects defines earlier.

13. If user selects option 2.

14. Get the account number from the user.

15. Find that account number in the array and shift the next element of the array at that position.

16. If user selects option 3.

17. Print the contents of array using for loop.

18. If user selects option 4.

19. Exit

**Implementation details:**

import java.util.Scanner;

class exp4 {

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

Scanner sc = *new* Scanner(System.in);

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

int n = sc.nextInt();

account arr[] = *new* account[n];

int t = 0;

*while* (t != 4) {

System.out.println("(1) Add Account");

System.out.println("(2) Delete Account");

System.out.println("(3) Display Accounts");

System.out.println("(4) Exit");

t = sc.nextInt();

*if* (t == 1)

addAccount(arr, n);

*else* *if* (t == 2) {

System.out.print("Enter Account ID. to Delete: ");

int id = sc.nextInt();

deleteAccount(arr, id, n);

} *else* *if* (t == 3){

System.out.println("Enter the account id to search: ");

int id=sc.nextInt();

displayAccount(arr, id,n);

}

*else* *if* (t == 4)

*break*;

*else*

System.out.println("Please Enter Correct Option!");

}

}

*static* void addAccount(*account* *arr*[], int *n*) {

Scanner sc = *new* Scanner(System.in);

System.out.print("Enter Account ID: ");

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter Account Name: ");

String name = sc.nextLine();

System.out.print("Enter Account Balance: ");

double balance = sc.nextDouble();

boolean check = false;

*for* (int i = 0; i < *n*; i++) {

*if*(*arr*[i]!=null){

*if*(*arr*[i].id==id){

check=true;

System.out.println("Cannot add the above account id");

*break*;

}

}

*else* {

*arr*[i] = *new* account(id, name, balance);

check = true;

*break*;

}

}

*if* (check == false)

System.out.println("There is not enough space in Array!");

}

*static* void deleteAccount(*account* *arr*[], int *id*, int *n*) {

boolean check = false;

*for* (int i = 0; i < *n* && *arr*[i]!=null; i++) {

*if* (*arr*[i].id == *id*) {

check = true;

*for* (int j = i; j < *n* - 1; j++)

*arr*[j] = *arr*[j + 1];

*arr*[*n* - 1] = null;

}

}

*if* (check == false)

System.out.println("Account ID not found!");

}

*static* void displayAccount(*account* *arr*[], int *id*,int *n*) {

boolean check=false;

*for*(int i=0;i<*n* && *arr*[i]!=null;i++){

*if*(*arr*[i].id==*id*){

System.out.println("\nAccount id: " + *arr*[i].id);

System.out.println("Account name: " + *arr*[i].name);

System.out.println("Account balance: " + *arr*[i].balance + "\n");

check=true;

*break*;

}

}

*if*(check==true)

System.out.println("Account Not Found!");

}

}

class account {

int id;

String name;

double balance;

account() {

id = 0;

name = null;

balance = 0;

}

account(int *pid*, String *pname*, double *pbalance*) {

id = *pid*;

name = *pname*;

balance = *pbalance*;

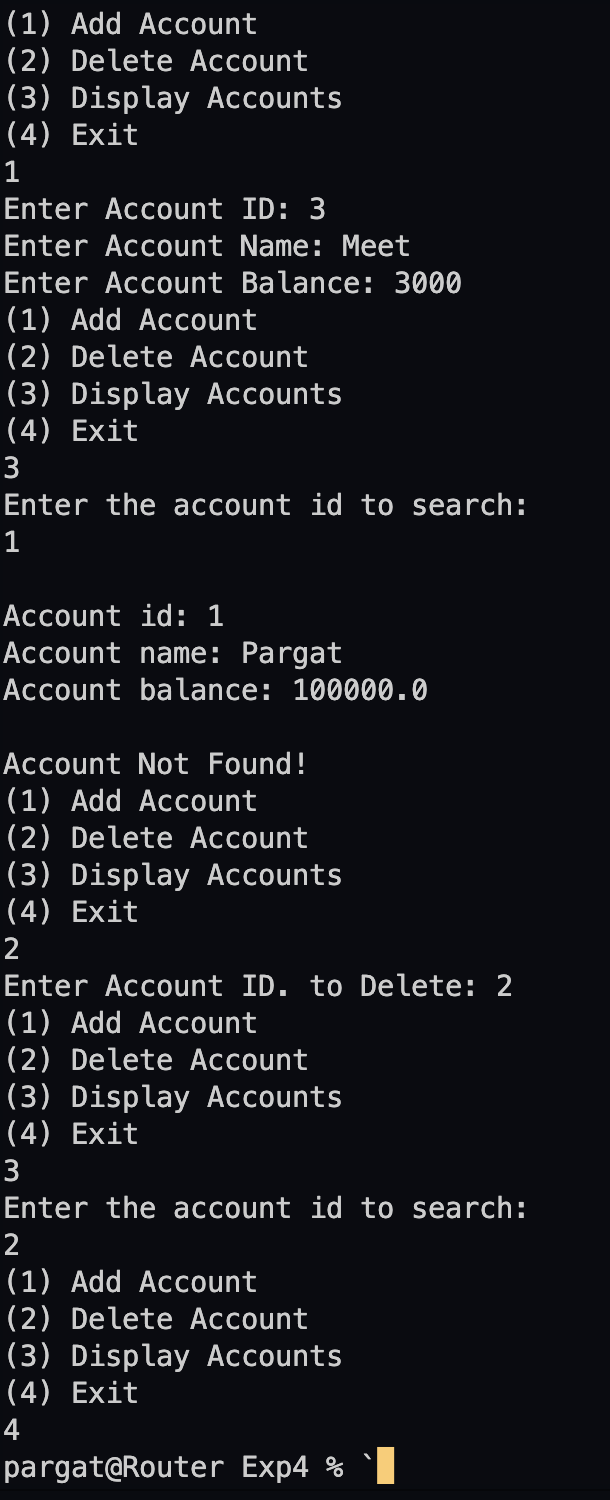
}

}

**Output:**

Text

Description automatically generated



**Conclusion: The experiment was executed successfully.**

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

**Q.1**  If an array of objects is of size 10 and a data value have to be retrieved from 5th object then \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ syntax should be used.

a)Array\_Name[4].data\_variable\_name;  
b)Data\_Type Array\_Name[4].data\_variable\_name;  
c)Array\_Name[4].data\_variable\_name.value;  
d) Array\_Name[4].data\_variable\_name(value);

**Ans: a**

 Q.2)The Object array is created in \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a)Heap memory  
b) Stack memory  
c) HDD  
d) ROM

**Ans: a**

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No.05**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :Vector** |

**AIM:** Create a class Employee which stores E-Name, E-Id and E-Salary of an Employee. Use class Vector to maintain an array of Employee with respect to the E-Salary. Provide the following functions

1) Create (): this function will accept the n Employee records in any order and will arrange them in the sorted order.

2) Insert (): to insert the given Employee record at appropriate index in the vector depending upon the E-Salary.

3) delete ByE-name( ): to accept the name of the Employee and delete the record having given name

4) deleteByE-Id ( ): to accept the Id of the Employee and delete the record having given E-Id.

Provide the following functions

1. boolean add(E e) : This method appends the specified element to the end of this Vector.
2. void addElement(E obj) This method adds the specified component to the end of this vector, increasing its size by one.
3. int lastIndexOf(Object o, int index) This method returns the index of the last occurrence of the specified element in this vector, searching backwards from index, or returns -1 if the element is not found.
4. void removeElementAt(int index)This method deletes the component at the specified index.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Vectors in Java are one of the most commonly used data structures. Similar to Arrays data structures which hold the data in a linear fashion. Vectors also store the data in a linear fashion, but unlike Arrays, they do not have a fixed size. Instead, their size can be increased on demand.

Vector class is a child class of AbstractList class and implements on List interface. To use Vectors, we first have to import Vector class from java.util package:

import java.util.Vector;

**Access Elements in Vector:**

We can access the data members simply by using the index of the element, just like we access the elements in Arrays.

Example- If we want to access the third element in a vector v, we simply refer to it as v[3].

**Vectors Constructors**

Listed below are the multiple variations of vector [constructors](https://www.edureka.co/blog/constructor-in-java/) available to use:

1. **Vector(int initialCapacity, int Increment)** – Constructs a vector with given initialCapacity and its Increment in size.
2. **Vector(int initialCapacity)*–***Constructs an empty vector with given initialCapacity. In this case, Increment is zero.
3. **Vector()** – Constructs a default vector of capacity 10.
4. **Vector(Collection c)*–***Constructs a vector with a given collection, the order of the elements is same as returned by the collection’s iterator.

There are also three protected parameters in vectors

* + **Int capacityIncrement()-** It automatically increases the capacity of the vector when the size becomes greater than capacity.
  + **Int elementCount()** – tell number of elements in the vector
  + **Object[] elementData()** – array in which elements of vector are stored

**Memory allocation of vectors:**

Vectors do not have a fixed size, instead, they have the ability to change their size dynamically. One might think that the vectors allocate indefinite long space to store objects. But this is not the case. Vectors can change their size based on two fields ‘capacity’ and ‘capacityIncrement’. Initially, a size equal to ‘capacity’ field is allocated when a vector is declared. We can insert the elements equal to the capacity. But as soon as the next element is inserted, it increases the size of the array by size ‘capacityIncrement’. Hence, it is able to change its size dynamically.

For a default constructor, the capacity is doubled whenever the capacity is full and a new element is to be inserted.

**Methods of Vectors :**

* Adding elements
* Removing elements
* Changing elements
* Iterating the vector

**Class Diagram:**

|  |  |
| --- | --- |
| Class name | Exp5 |
| Variables | - |
| Functions | main(), addAccount( Vector<Employee> arr, int n),  deleteAccount(Vector<Employee> arr),  displayAccount(Vector<Employee> arr) |

|  |  |
| --- | --- |
| Class name | Employee |
| Variables | Int id, String name, double salary |
| Functions | - |

**Algorithm:**

1. Start

2. Create a Vector object Emp of the type Employee(E\_Name, E\_Id, E\_Salary)

3. while(true)

4. 1.Create 2.Insert by salary 3.Delete by name 4.Delete by Id 5.Display 6. Exit

5. read choice

6. (Switch Case), Case 1(choice=1)

6.1 read number of employee records to be added(n)

6.2 for i=0, i<n, accept the employee details(e\_name. e\_id, e\_salary)

6.3 Sort the employee records using comparator interface

6.3.1 if a.E\_Salary>b.E\_Salary, return 1

6.3.2 else if a.E\_Salary<b.E\_Salary, return -1

6.3.3 else, return 0

7. Case 2(choice=2)

7.1 accept the employee details

7.2 for i=0, i<Emp.size()

If Emp.get(i).E\_Salary>e\_salary, add the record to the i

th index.

8. Case 3(choice=3)

8.1 Declare a=0

8.2 read the name of record to be deleted 8.3 for i=0, i<Emp.size()

if e\_name = Emp.get(i).E\_Name, remove the corresponding record and increment a

8.4 if a=0, print “Employee name not found.”

9. Case 4(choice=4)

9.1 Declare b=0

9.2 read the name of record to be deleted 9.3 for i=0, i<Emp.size()

if e\_id = Emp.get(i).E\_Id, remove the corresponding record and increment b

9.4 if b=0, print “Employee id not found.”

10. Case 5(choice=5)

Print all the records

11. Case 6(choice=6)

12. Exit

**Implementation details:**

import java.util.\*;

class exp5 {

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

Scanner sc = *new* Scanner(System.in);

Vector<Employee> arr = *new* Vector<>();

int t = 0;

System.out.print("Enter the number of Employees to add: ");

int n = sc.nextInt();

addAccount(arr, n);

*while* (t != 4) {

System.out.println("(1) Add Employee");

System.out.println("(2) Delete Employee");

System.out.println("(3) Display Employee");

System.out.println("(4) Exit");

t = sc.nextInt();

*if* (t == 1)

addAccount(arr, 1);

*else* *if* (t == 2)

deleteAccount(arr);

*else* *if* (t == 3)

displayAccount(arr);

*else* *if* (t == 4)

*break*;

*else*

System.out.println("Please Enter Correct Option!");

}

}

*static* void addAccount(Vector<Employee> *arr*, int *n*) {

Scanner sc = *new* Scanner(System.in);

*for* (int i = 0; i < *n*; i++) {

System.out.print("Enter Employee ID: ");

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter Employee Name: ");

String name = sc.nextLine();

System.out.print("Enter Employee salary: ");

double balance = sc.nextDouble();

int check = 0;

*for* (Employee e *:* *arr*) {

*if* (e.id == id) {

System.out.println("Cannot add the above Employee data!");

check = -1;

*break*;

} *else* *if* (e.name.equals(name)) {

System.out.println("Cannot add the above Employee data!");

check = -1;

*break*;

}

}

*if* (check != -1)

*arr*.add(*new* Employee(id, name, balance));

}

Collections.sort(*arr*, *new* Comparator<Employee>() {

*public* int compare(Employee *s1*, Employee *s2*) {

*if* (*s1*.salary > *s2*.salary)

*return* -1;

*else* *if* (*s1*.salary < *s2*.salary)

*return* 1;

*else*

*return* 0;

}

});

}

*static* void deleteAccount(Vector<Employee> *arr*) {

Scanner sc = *new* Scanner(System.in);

System.out.println("(1) Delete by ID");

System.out.println("(2) Delete by Name");

int t = sc.nextInt();

int id = -1, count = 0;

String name = null;

*if* (t == 1) {

System.out.println("Enter Id");

id = sc.nextInt();

} *else* *if* (t == 2) {

System.out.println("Enter Name");

sc.nextLine();

name = sc.nextLine();

}

*for* (Employee e *:* *arr*) {

*if* (e.id == id && t == 1) {

*arr*.remove(e);

System.out.println("Deleted Employee!");

*break*;

} *else* *if* (e.name.equals(name) && t == 2) {

*arr*.remove(e);

System.out.println("Deleted Employee!");

*break*;

}

count++;

}

}

*static* void displayAccount(Vector<Employee> *arr*) {

*for* (Employee employee *:* *arr*) {

System.out.println("\n\nName: " + employee.name);

System.out.println("Id: " + employee.id);

System.out.println("Salary: " + employee.salary);

}

}

}

class Employee {

int id;

String name;

double salary;

Employee() {

id = 0;

name = null;

salary = 0;

}

Employee(int *pid*, String *pname*, double *pbalance*) {

id = *pid*;

name = *pname*;

salary = *pbalance*;

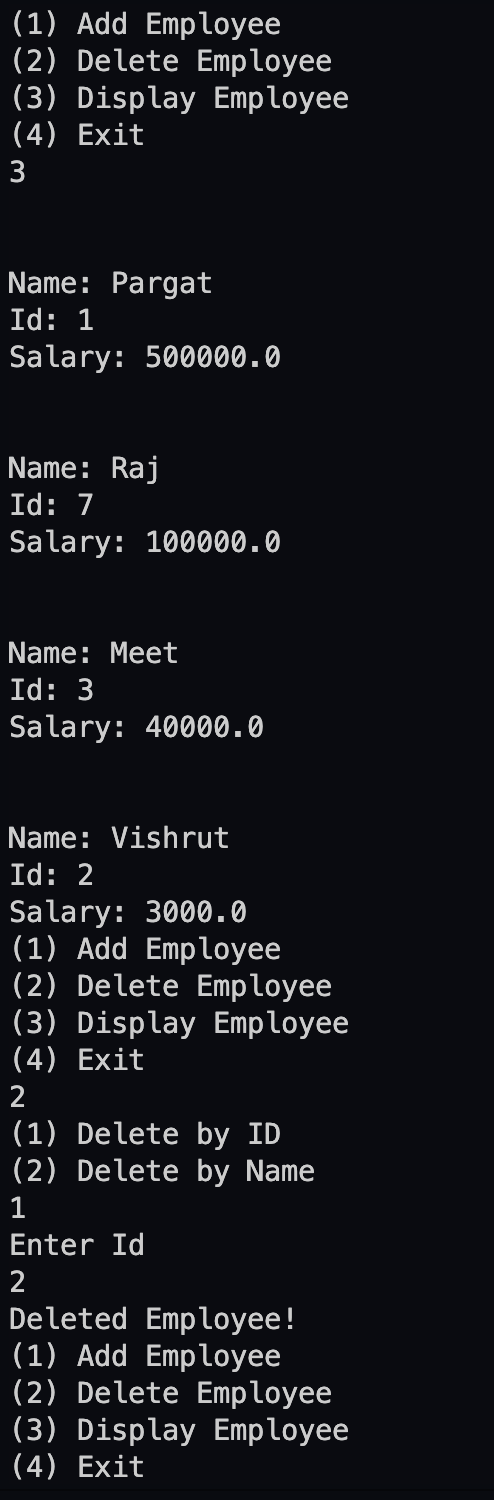
}

}

**Output:**

Text

Description automatically generated

Text

Description automatically generated

**Conclusion:**

The concept of vectors in Java was studied and implemented.

**Date:\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

1. **What is the output of the following Program**

|  |
| --- |
| import java.util.\*;  class demo2 {      public static void main(String[] args)      {          Vector v = new Vector(20);          v.addElement("Geeksforgeeks");          v.insertElementAt("Java", 2);          System.out.println(v.firstElement());      }  } |

**Output**: It gives no output because of ArrayindexOutOfBound Exception

because initially there was only one element at index 0. Next it expects a value to

be added at index 1, but we are adding at index 2. Hence an exception is raised.

1. **Expain any 10 methods of Vector class in detail with the help of example**

**1. void add(int index, Object element)**

Inserts the specified element at the specified position in this Vector.

Example: v.add(1, “apple”);

**2. boolean add(Object o)**

Appends the specified element to the end of this Vector.

Example: v.add(“pear”);

**3. boolean addAll(int index, Collection c)**

Inserts all of the elements in the specified Collection into this Vector at the

specified position.

Example: v.addAll(3, arr);

**4. void addElement(Object obj)**

Adds the specified component to the end of this vector, increasing its size

by one.

Example: v.addElement(“plum”);

**5. int capacity()**

Returns the current capacity of this vector.

Example: v.capacity();

**6. boolean contains(Object elem)**

Tests if the specified object is a component in this vector.

Example: v.contains(“apple”);

**7. void copyInto(Object[] anArray)**

Copies the components of this vector into the specified array.

Example: v.copyInto(arr);

**8. Object elementAt(int index)**

Returns the component at the specified index.

Example: v.elementAt(2);

**9. Object remove(int index)**

Removes the element at the specified position in this vector.

Example: v. remove(3);

**10.int size()**

Returns the number of components in this vector.

Example: v.size();

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No.06**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :Case Study (for Class Diagram)** |

**AIM:** Draw class Diagram for the chosen Case Study . Clearly show

* + Attributes
  + Multiplicities between classes
  + Aggregations/compositions/Association between classes
  + Generalization between classes in the class diagram.

And show the implementation of aggregation, association, composition and generalization between the classes.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1:** Understand the features of object oriented programming compared with procedural approach with C++ and Java.

**CO2**: Explore arrays, vectors, classes and objects in C++ and Java.

**CO3:** Implement scenarios using object oriented concepts (Drawing class diagram, relationship between classes, sequence diagram)

**CO4**: Explore the interface, exceptions, multithreading, packages

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1.Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

DefineClass, Methods, Object.

Understanding of Aggregation, Association, Composition and Generalization between classes

**List Of Classes:**

**Staff**: Contains the staff details

**Doctor**: Contains the details of doctors

**Patient**: Contains the patients details

**Medical**: Contains medicine details

**Facility**: Contains details about facilities available

**Identify Attributes for each class:**

**Staff:** sid, sname, desg, gender

**Doctor:** did, dname, specilist, doc\_qual, droom

**Patient:** pid, pname, disease, gender, admit\_status, age

**Medical:** med\_name, exp\_date, med\_cost, count

**Facility:** fac\_name

**Lab:** facility, lab\_cost

**Identify List of Methods in each classes:**

Staff: new\_staff(id,name,deg,gender,salary), staff\_info(id), staff\_del()

Doctor: new\_doctor(did, dname, specilist, doc\_qual, droom), doctor\_info(), doctor\_del()

Patient: new\_patient(), patient\_info(id), patient\_del()

Medical: new\_medi(), medi\_info(id),medical\_delete()

Facility: add\_faci(), show\_faci(id),facility\_delete()

**Class Diagram:**

Diagram

Description automatically generated

**Algorithm:**

1. Start
2. import java.util.\* and java.util.calendar
3. Declare a class Staff
4. Declare attributes of class Staff sid, sname, desg, sex, salary.
5. Declare a method new\_staff in Staff class to set the attributes for new members of the class.
6. Declare a method staff\_info to print these details.
7. Declare a class Doctor
8. Declare attributes of class Doctor did, dname, specilist, appoint, doc\_qual, droom
9. Declare method new\_doctor and doctor\_jnfo which has same purpose as class Staff methods had for Staff.
10. In the same way as done in previous steps declare classes Patient, Medical, Facility, Lab
11. Declare following attributes for the respective classes

Patient: pid, pname, disease, sex, admit\_status, age  
Medical: med\_name, med\_comp, exp\_date, med\_cost, count Facility: fac\_name  
Lab: facility, lab\_cost.

1. Now, declare following methods for the classes and these methods serve the same purpose as described for class Staff and Doctor i.e. to set the attributes for new members of the class and to print the details.

Patient: new\_patient(), patient\_info() Medical: new\_medi()  
Facility: add\_faci(), show\_faci() Lab: new\_faci(), faci\_list()

1. In the main method define a String containing months names
2. Create a object calendar for calendar.
3. Declare variables count1, count2, count3, count4, count5,count6.
4. Print current date and time.
5. Create Array of objects for the respective classes.
6. Initialize the for loops for the classes which are used to add new objects to the class.
7. Add some objects to this array list.
8. Get choice from the user.
9. If 1 is selected then user gets two options 1. Add New Entry 2. Existing Doctorslist
10. After to go back enter 1 and 0 for main menu.
11. According to choose entered by the user add the objects to the class or display the details using the methods defined earlier.
12. Repeat step 22 and 23 for all other options.
13. If user enters 1.
14. Exit

**Conclusion : Successfully completed the given task.**

**Date: \_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

**1.** **Consider the following class:**

public class TypeOfVariable{

public static int a;

int b,c;

public void printValue(){

int x = 10;

}

public static void main(String args[]){

TypeOfVariable object=new TypeOfVariable();

object.printValue();

}

}

**a). What are the class/static variables?**

A static variable is common to all the instances (or objects) of the class because it is a class level variable. In other words you can say that only a single copy of static variable is created and shared among all the instances of the class. Memory allocation for such variables only happens once when the class is loaded in the memory. Static variables are also known as Class Variables.  
Unlike non-static variables, such variables can be accessed directly in static and non-static methods.

Here **a** is static variable.

**b). What are the instance variables?**

Instance variable in Java is used by Objects to store their states. Variables which are defined without the STATIC keyword and are Outside any method declaration are Object-specific and are known as instance variables. They are called so because their values are instance specific and are not shared among instances.

Here **b** and **c** are instance variables.

**c.)What are local variables?**

A local variable in Java is a variable that's declared within the body of a method. Then you can use the variable only within that method. Other methods in the class aren't even aware that the variable exists. You don't specify static on a declaration for a local variable.

Here **x** is a local variable.

**2.What is the output from the following code:**

public class Test

{

    static int x = 11;

    private int y = 33;

    public void method1(int x)

    {

        Test t = new Test();

        this.x = 22;

        y = 44;

        System.out.println("Test.x: " + Test.x);

        System.out.println("t.x: " + t.x);

        System.out.println("t.y: " + t.y);

        System.out.println("y: " + y);

    }

    public static void main(String args[])

    {

        Test t = new Test();

        t.method1(5);

    }

}

**Output**

Test.x = 22 t.x = 22  
t.y = 33  
y = 44

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No.07**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE : User Defined Exception** |

**AIM:**

Create a user defined exception subclass TimeException with necessary constructors and overridden toString method. Write a program which accepts two integers with time in minutes and seconds and find the sum. It throws an object of the TimeException class if the value exceeds 60seconds otherwise it displays the total time. On printing, the exception object should display an exception name, appropriate message for exception.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1:** Understand the features of object oriented programming compared with procedural approach with C++ and Java

**CO4:**Explore the interface, exceptions, multithreading, packages **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1.Ralph Bravaco , Shai Simoson , “Java Programming From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

**Exception handling** in java is a powerful mechanism or technique that allows us to handle runtime errors in a program so that the normal flow of the program can be maintained. All the exceptions occur only at runtime. A syntax error occurs at compile time.

**Exception in Java:**

In general, an exception means a problem or an abnormal condition that stops a computer program from processing information in a normal way.

An exception in java is an object representing an error or an abnormal condition that occurs at runtime execution and interrupts (disrupts) the normal execution flow of the program.

An exception can be identified only at runtime, not at compile time. Therefore, it is also called runtime errors that are thrown as exceptions in Java. They occur while a program is running.

For example:

* If we access an array using an index that is out of bounds, we will get a runtime error named ArrayIndexOutOfBoundsException.
* If we enter a double value while the program is expecting an integer value, we will get a runtime error called InputMismatchException.

When JVM faces these kinds of errors or dividing an integer by zero in a program, it creates an exception object and throws it to inform us that an error has occurred.If the exception object is not caught and handled properly, JVM will display an error message and will terminate the rest of the program abnormally.

If we want to continue the execution of remaining code in the program, we will have to handle exception objects thrown by error conditions and then display a user-friendly message for taking corrective actions. This task is known as exception handling in java.

**Types of Exceptions in Java**

Basically, there are two types of exceptions in java API. They are:

1. Predefined Exceptions (Built-in-Exceptions)

2. Custom (User defined)Exceptions

**Predefined Exceptions:**

Predefined exceptions are those exceptions that are already defined by the Java system. These exceptions are also called built-in-exceptions.Java API supports exception handling by providing the number of predefined exceptions. These predefined exceptions are represented by classes in java.

When a predefined exception occurs, JVM (Java runtime system) creates an object of predefined exception class. All exceptions are derived from java.lang.Throwable class but not all exception classes are defined in the same package. All the predefined exceptions supported by java are organized as subclasses in a hierarchy under the Throwable class.

All the predefined exceptions are further divided into two groups:

1. Checked Exceptions: Checked exceptions are those exceptions that are checked by the java compiler itself at compilation time and are not under runtime exception class hierarchy. If a method throws a checked exception in a program, the method must either handle the exception or pass it to a caller method.

2. Unchecked Exceptions: Unchecked exceptions in Java are those exceptions that are checked by JVM, not by java compiler. They occur during the runtime of a program. All exceptions under the runtime exception class are called unchecked exceptions or runtime exceptions in Java.

**Custom exceptions:**

Custom exceptions are those exceptions that are created by users or programmers according to their own needs. The custom exceptions are also called user-defined exceptions that are created by extending the exception class.

So, Java provides the liberty to programmers to throw and handle exceptions while dealing with functional requirements of problems they are solving.

**Exception Handling Mechanism using Try-Catch block:**

The general syntax of try-catch block (exception handling block) is as follows:

**Syntax:**

try

{

// A block of code; // generates an exception

}

catch(exception\_class var)

{

// Code to be executed when an exception is thrown.

}

**Example:**

public class TryCatchEx

{

public static void main(String[] args)

{

System.out.println("11");

System.out.println("Before divide");

int x = 1/0;

System.out.println("After divide");

System.out.println("22");

}

}

Output**:**

11

Before divide

Exception in thread "main" java.lang.ArithmeticException: / by zero

**Class Diagram:**

**Algorithm:**

Step 1: Create a class that extends the class Exception and call the constructor of that class.

Step 2: In the constructor initialize the class variables.

Step 3: In the main method of class Exp7 input the time from the user and check if they

are valid in the try block.

Step 4: If the time is invalid then throws an exception .In the catch block print the

exception name and the message that needs to be printed along with it.

Step 5: If the time entered is vaild then display the total seconds.

**Implementation details :**

import java.util.Scanner;

class TimeException extends Exception {

String msg;

*public* TimeException(String *s*) {

this.msg = *s*;

}

*public* String toString() {

*return* (msg);

}

}

*public* class Exp7 {

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

Scanner sc = *new* Scanner(System.in);

*try* {

System.out.print("Enter minutes: ");

int min = sc.nextInt();

System.out.print("Enter seconds: ");

int sec = sc.nextInt();

*if* (sec > 60)

*throw* *new* TimeException("seconds can't be more than 60 secs");

*else* {

sec=60\*min+sec;

System.out.println("Total Seconds is "+sec+"sec");

}

} *catch* (TimeException *e*) {

System.out.print("TimeException Error: ");

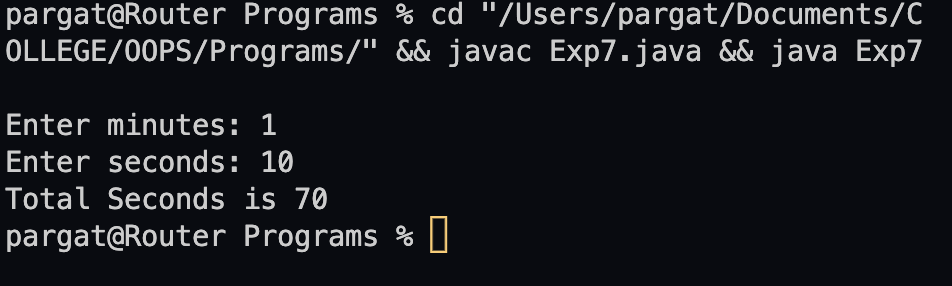
System.out.println(e.toString());

}

}

}

**Output:**



Text

Description automatically generated with medium confidence

**Conclusion:** User defined exceptions were understood and implemented successfully.

**Date: \_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

1. **Compare throw and throws.**

|  |  |
| --- | --- |
| **throw** | **throws** |
| The throw keyword is used inside a function. It is used when it is required to throw an Exception logically. | The throws keyword is used in the function signature. It is used when the function has some statements that can lead to exceptions. |
| The throw keyword is used to throw an exception explicitly. It can throw only one exception at a time. | The throws keyword can be used to declare multiple exceptions, separated by a comma. Whichever exception occurs, if matched with the declared ones, is thrown automatically then. |
| Syntax of throw keyword includes the instance of the Exception to be thrown. Syntax wise throw keyword is followed by the instance variable. | Syntax of throws keyword includes the class names of the Exceptions to be thrown. Syntax wise throws keyword is followed by exception class names. |
| throw keyword cannot propagate checked exceptions. It is only used to propagate the unchecked Exceptions that are not checked using the throws keyword. | throws keyword is used to propagate the checked Exceptions only. |

1. **Explain how to create a user defined exception and explicitly throw an exception in a program with a simple example.**

Java user-defined exception is a custom exception created and throws that exception using a keyword ‘throw’. It is done by extending a class ‘Exception’. An exception is a problem that arises during the execution of the program. In Object-Oriented Programming language, Java provides a powerful mechanism to handle such exceptions. Java allows to create own exception class, which provides own exception class implementation. Such exceptions are called user-defined exceptions or custom exceptions.

Example:

class Example {

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

*try* {

*throw* *new* NewException(100);

} *catch* (NewException *e*) {

System.out.println(e);

}

}

}

class NewException extends Exception {

int a;

NewException(int *b*) {

a = *b*;

}

*public* String toString() {

*return* ("Status code = " + a);

}

}

1. **Suppose the statement2 causes an exception in following try-catch block:**

try {

statement1;

statement2;

statement3;

}

catch(Exception1 e1) {

}

catch(Exception2 e2){

}

statement4;

Answer the following questions:

* Will statement3 be executed?

**NO, the statement 3 will not be executed.**

* If the exception is not caught, will statement4 be executed?

**NO, the statement 4 will not be executed.**

* If the exception is caught in the catch block, will statement4 be executed?

**YES, if the exception is caught in the catch block, statement 4 will be executed.**

* If the exception is passed to the caller, will the statement4 be executed?

**YES, it the exception is passed to the caller, statement 4 will be executed.**

1. Explain finally block with the help of an example.

The finally block in java is used to put important codes such as clean up code e.g. closing the file or closing the connection. The finally block executes whether exception rise or not and whether exception handled or not. A finally contains all the crucial statements regardless of the exception occurs or not.

Example:

class Test {

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

*try* {

int data = 25 / 5;

System.out.println(data);

} *catch* (NullPointerException *e*) {

System.out.println(e);

} *finally* {

System.out.println("finally block is always executed");

}

System.out.println("rest of the code...");

}

}

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No.08**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE : Multithreading Programming** |

**AIM:** Write a java program that implements a multi-thread application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1:** Understand the features of object oriented programming compared with procedural approach with C++ and Java

**CO4:** Explore the interface, exceptions, multithreading, packages.

. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Ralph Bravaco , Shai Simoson , “Java Programming From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Java provides built-in support for multithreaded programming. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution. A multithreading is a specialized form of multitasking. Multithreading requires less overhead than multitasking processing.

Multithreading enables you to write very efficient programs that make maximum use of the CPU, because idle time can be kept to a minimum.

**Creating a Thread:**

Java defines two ways in which this can be accomplished:

1. You can implement the Runnable interface.
2. You can extend the Thread class itself.

**Create Thread by Implementing Runnable:**

The easiest way to create a thread is to create a class that implements the Runnable interface.

To implement Runnable, a class needs to only implement a single method called run( ), which is declared like this:

public void run( )

You will define the code that constitutes the new thread inside run() method. It is important to understand that run() can call other methods, use other classes, and declare variables, just like the main thread can.

After you create a class that implements Runnable, you will instantiate an object of type Thread from within that class. Thread defines several constructors. The one that we will use is shown here:

Thread(Runnable threadOb, String threadName);

Here, threadOb is an instance of a class that implements the Runnable interface and the name of the new thread is specified by threadName.

After the new thread is created, it will not start running until you call its start( ) method, which is declared within Thread. The start( ) method is shown here:

void start( );

Here is an example that creates a new thread and starts it running:

class NewThread implements Runnable {

Thread t;

NewThread() {

t = new Thread(this, "Demo Thread");

System.out.println("Child thread: " + t);

t.start(); // Start the thread

}

public void run() {

try {

for(int i = 5; i > 0; i--) {

System.out.println("Child Thread: " + i);

// Let the thread sleep for a while.

Thread.sleep(50);

}

} catch (InterruptedException e) {

System.out.println("Child interrupted.");

}

System.out.println("Exiting child thread.");

}

}

public class ThreadDemo {

public static void main(String args[]) {

new NewThread();

try {

for(int i = 5; i > 0; i--) {

System.out.println("Main Thread: " + i);

Thread.sleep(100);

}

} catch (InterruptedException e) {

System.out.println("Main thread interrupted.");

}

System.out.println("Main thread exiting.");

}

}

The second way to create a thread is to create a new class that extends Thread, and then to create an instance of that class.

The extending class must override the run( ) method, which is the entry point for the new thread. It must also call start( ) to begin execution of the new thread.

class NewThread extends Thread {

NewThread() {

super("Demo Thread");

System.out.println("Child thread: " + this);

start(); // Start the thread

}

public void run() {

try {

for(int i = 5; i > 0; i--) {

System.out.println("Child Thread: " + i);

// Let the thread sleep for a while.

Thread.sleep(50);

}

} catch (InterruptedException e) {

System.out.println("Child interrupted.");

}

System.out.println("Exiting child thread.");

}

}

public class ExtendThread {

public static void main(String args[]) {

new NewThread(); // create a new thread

try {

for(int i = 5; i > 0; i--) {

System.out.println("Main Thread: " + i);

Thread.sleep(100);

}

} catch (InterruptedException e) {

System.out.println("Main thread interrupted.");

}

System.out.println("Main thread exiting.");

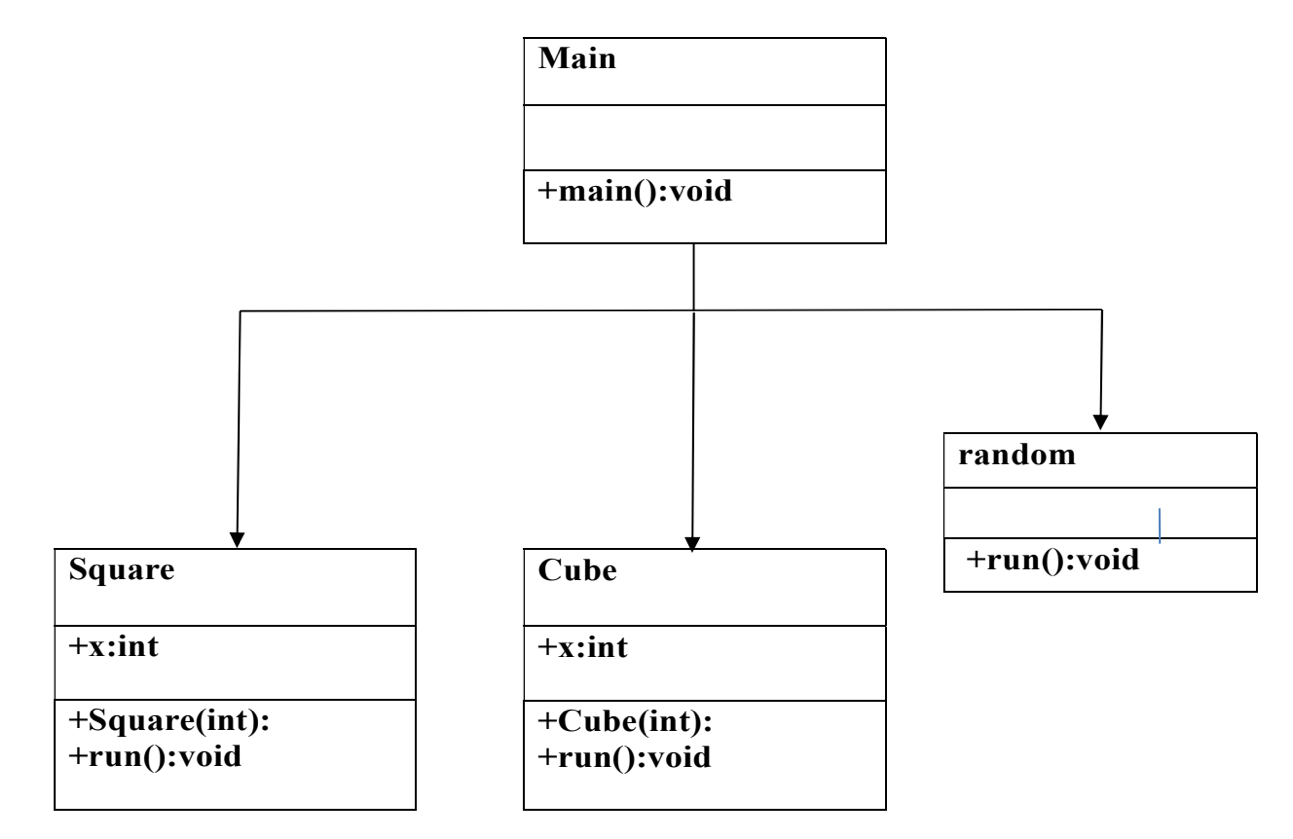
}

}

**Some of the Thread methods**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| void setName(String name) | Changes the name of the Thread object. There is also a getName() method for retrieving the name |
| Void setPriority(int priority) | Sets the priority of this Thread object. The possible values are between 1 and 10. 5 |
| boolean isAlive() | Returns true if the thread is alive, which is any time after the thread has been started but before it runs to completion. |
| void yield() | Causes the currently running thread to yield to any other threads of the same priority that are waiting to be scheduled. |
| void sleep(long millisec) | Causes the currently running thread to block for at least the specified number of milliseconds. |
| Thread currentThread() | Returns a reference to the currently running thread, which is the thread that invokes this method. |

**Class Diagram:**



**Algorithm:**

Step 1: Create class random and extend Thread to it

Step 2: Override the function run

i. In function run generate a random number by using random package

ii. Create objects of thread Square and thread cube

iii. Check if the number generated is even, if so , then start thread Square else

start thread Cube

Step 3: In class Square, create a parametrized constructor to initialize the class variables

Step 4: Override the function run , which prints the square of the number.

Step 5: In class Cube, create a parametrized constructor to initialize the class variables

Step 6: Override the function run , which prints the cube of the number .

Step 7: Create a class Main , with a main function.

Step 8: The main function is used to create an object of class Random .This object is

then used to start the Random thread.

**Implementation details:**

import java.util.Random;

class random extends Thread {

*public* void run() {

Random r = *new* Random();

*for* (int i = 0; i < 5; i++) {

int x = r.nextInt(100);

*try* {

Thread.sleep(1000);

} *catch* (Exception *e*) {

} *finally* {

Square obj2 = *new* Square(x);

Cube obj3 = *new* Cube(x);

*if* (x % 2 == 0)

obj2.start();

*else*

obj3.start();

}

}

}

}

class Square extends Thread {

*public* int x;

*public* Square(int *x*) {

this.x = *x*;

}

*public* void run()

{

System.out.println("Random Number is "+x+", Square is: "+x\*x);

}

}

class Cube extends Thread {

int x;

Cube(int *x*) {

this.x = *x*;

}

*public* void run()

{

System.out.println("Random Number is "+x+", Cube is : "+x\*x\*x);

}

}

*public* class exp8 {

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

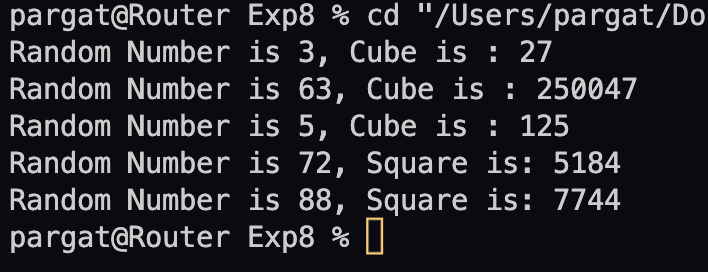
random obj1 = *new* random();

obj1.start();

}

}

**Output:**



**Conclusion:**

Thus, multithreading was understood and implemented.

**Date:\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

1. **What do you mean by multithreading?**

Multithreading in Java is a process of executing two or more threads

simultaneously to maximum utilization of CPU. Multithreaded applications

execute two or more threads run concurrently. Hence, it is also known as

Concurrency in Java. Each thread runs parallel to each other. Multiple threads

don’t allocate separate memory area, hence they save memory. Also, context

switching between threads takes less time.

**2. Explain the use of sleep and run function with an example?**

Thread.sleep() method can be used to pause the execution of current thread for

specified time in milliseconds. The argument value for milliseconds can’t be negative,

else it throws IllegalArgumentException.

Example :

public class ThreadSleep {

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

long start = System.currentTimeMillis();

Thread.sleep(2000);

System.out.println("Sleep time in ms = " + (System.currentTimeMillis() - start));

}}

The run() method of thread class is called if the thread was constructed using a separate

Runnable object otherwise this method does nothing and returns. When the run()

method calls, the code specified in the run() method is executed. You can call the run()

method multiple times

public class RunExp1 implements Runnable {

*public* void run() {

System.out.println("Thread is running...");

}

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

RunExp1 r1 = *new* RunExp1();

Thread t1 = *new* Thread(r1);

*// this will call run() method*

t1.start();

}

}

**3. Explain any five methods of Thread class with Example ?**

activeCount() : Returns an estimate of the number of active threads in the current thread’s thread group and its subgroups.

checkAccess() : Determines if the currently running thread has permission to modify this thread.

Clone(): Throws CloneNotSupportedException as a Thread can not be meaningfully cloned

currentThread() : Returns a reference to the currently executing thread object

dumpStack(): Prints a stack trace of the current thread to the standard error stream

class MyThread extends Thread {

*public* void run() {

System.out.println("Thread is running created by extending to parent Thread class");

}

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

MyThread myThread = *new* MyThread();

myThread.start();

}

}

**Batch: A3 Roll No.: 16010121045**

**Experiment / assignment / tutorial No. 9**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :Java Packages** |

**AIM:** Create a **Package Engineering** which has two classes as **Student and Marks**. Accept (n) student details like roll\_no, Subject\_name, Student\_name,calculate total marks in the class Student Write **display () method** to display details and **sort () method** to sort the students records as per increasing order of the total marks. The function **sort must be statically defined to invoke it without referring to any object**. Both the functions are written in the Marks class.

Create a main class which will use a package to display all the records of the student in the increasing order of their total marks.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO4:** Explore the interface, exceptions, multithreading, packages.

**Books/ Journals/ Websites referred:**

1. Ralph Bravaco , Shai Simoson , “Java Programming From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

**Java Packages:**

A package in Java is a group of similar types of classes, interfaces, and sub-packages. They can be categorized into two categories, the built-in package ( java, lang, util, awt, javax, swing, net, io, sql et), and user-defined package.

They are used for the following tasks –

* To prevent the naming conflicts which can occur between the classes.
* Make the searching and locating of classes or enumerations or annotations much easier.
* Provide access control to the classes.
* Used for data encapsulation.

**Advantages of Java Package:**

* A Java package is mainly used for the categorization of classes and interfaces so that we can maintain them easily.
* They always provide access protection
* Used to bundle classes and interfaces.
* With the help of packages, we can reuse the existing code
* By using the package, we can easily locate the classes related to it.
* Also, remove the naming collision.

**Built-in Packages in Java**

Built-in is a part of Java API and it offers a variety of packages are –

lang – Automatically imported and it contains language support classes.

io – Contains classes for input and output operations.

util – Contains utility classes for implementing data structures.

applet – This package contains classes that create applets.

awt – Contain classes that implement compounds for GUI.

net – This package contains classes that support networking operations.

**User-defined Packages in Java**

1. package First;
2. public class MyClass
3. {
4. public void **getNames**(String name)
5. {
6. System.out.**println**(name);
7. }
8. }
9. package First;
10. import First.MyClass;
11. public class MyClass1 {
12. public static void **main**(String args[])
13. {
14. // Initializing the String variable with a value
15. String name = "Welcome";
16. // Creating an instance of class MyClass in the package.
17. MyClass obj = new **MyClass**();
18. obj.**getNames**(name);
19. }
20. }

.

**Class Diagram:**

|  |  |
| --- | --- |
| **CLASS NAME** | **MAIN** |
| **VARIABLES NAME** | +name: String  +n: int  +rno: int  +eng: int  +math: int  +sci: int  +total\_marks : int |
| **METHODS** | +main (): void |

|  |  |
| --- | --- |
| **CLASS NAME** | **Student** |
| **VARIABLES NAME** | +Student\_name: String  + roll\_no:int  +English: int  +Maths:int  +Science:int  +total\_marks:int  +e:int  +m:int  +s:int |
| **METHODS** | +cal(int , int ): int  +Student (int, int, int, int ,String): |
| **CLASS NAME** | **Marks** |
| **VARIABLES NAME** | +Student\_name : String  + roll\_no:int +English: int  +Maths:int  +Science:int  +total\_marks:int  +n:int |
| **METHODS** | +sort (int , int, int , int, String , int , int):void +display():void |

**Algorithm:**

Step 1: Create a package ‘Engineering’ having two classes: Student and Marks

Step 2: Class Student –

1. cal Function: Calculates total marks and returns it

Class Marks -

a.) sort Function: Sorts student information according to total marks obtained. If i+1 th student (marks) < i th student(marks) swap (continue process till n-1 iteration).

b.) display function: Displays student information

Step 3 Create MAIN class and import package ‘Engineering

Step 4: In main function accept number of students and their information.

Step 5: Store information of students in different arrays.

Step 6: call cal function and calculate total marks

Step 7: call sort function and then display the information using display function

**Implementation details:**

Main

import Engineering.\*;

*public* class Main {

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

Student student = *new* Student ();

Student sorted = *new* Student (student.RollNo.length);

student.getDetails();

student.calculateTotal();

sorted = Marks.sort(student);

Marks.display(sorted);

}

}

Marks

package Engineering;

*public* class Marks {

*public* *static* Student sort (Student *students*)

{

Student sorted = *new* Student (*students*.MarksTotal.length);

double toSort [] = *students*.MarksTotal;

int indexArr [] = *new* int [*students*.MarksTotal.length];

double min;

int minInd = 0;

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

indexArr[i] = i;

}

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

min = toSort[i];

minInd = i;

*// find minInd*

*for* (int j = i; j < indexArr.length; j++) {

*if* (toSort[j] < min) {

minInd = i;

}

}

*// swap minInd with i*

int temp = indexArr[i];

indexArr[i] = indexArr[minInd];

indexArr[minInd] = temp;

}

*// now sort*

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

int ind = indexArr[i];

*// move students[ind] to sorted[i]*

sorted.RollNo[i] = *students*.RollNo[ind];

sorted.StudentName[i] = *students*.StudentName[ind];

sorted.MarksChemistry[i] = *students*.MarksChemistry[ind];

sorted.MarksMaths[i] = *students*.MarksMaths[ind];

sorted.MarksPhysics[i] = *students*.MarksPhysics[ind];

sorted.MarksTotal[i] = *students*.MarksTotal[ind];

}

*return* sorted;

}

*public* *static* void display (Student *students*)

{

System.out.println("\nDetails of students");

*for* (int i = 0; i < *students*.RollNo.length; i++) {

System.out.println("\nStudent "+(i+1));

System.out.println("Student Roll Number: " + *students*.RollNo[i]);

System.out.println("Student Name: " + *students*.StudentName[i]);

System.out.println("Marks in Physics: " + *students*.MarksPhysics[i]);

System.out.println("Marks in Chemistry: " + *students*.MarksChemistry[i]);

System.out.println("Marks in Maths: " + *students*.MarksMaths[i]);

System.out.println("Total Marks: " + *students*.MarksTotal[i]);

}

}

}

Student

package Engineering;

import java.util.Scanner;

*public* class Student {

Scanner scanner = *new* Scanner (System.in);

*public* int RollNo [];

*public* String StudentName[];

*public* int MarksMaths[];

*public* int MarksPhysics [];

*public* int MarksChemistry [];

*public* double MarksTotal [];

*public* Student ()

{

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

int n = scanner.nextInt();

RollNo = *new* int [n];

StudentName = *new* String [n];

MarksMaths = *new* int [n];

MarksPhysics = *new* int [n];

MarksChemistry = *new* int [n];

MarksTotal = *new* double [n];

}

*public* Student (int *n*)

{

RollNo = *new* int [*n*];

StudentName = *new* String [*n*];

MarksMaths = *new* int [*n*];

MarksPhysics = *new* int [*n*];

MarksChemistry = *new* int [*n*];

MarksTotal = *new* double [*n*];

}

*public* void getDetails ()

{

*for* (int i = 0; i < RollNo.length; i++)

{

System.out.println ("\nDetails for student " + (i+1));

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

RollNo[i] = scanner.nextInt();

System.out.print ("Enter name of the student: ");

String trash = scanner.nextLine();

StudentName[i] = scanner.nextLine();

System.out.print ("Enter marks in maths: ");

MarksMaths[i] = scanner.nextInt();

System.out.print ("Enter marks in physics: ");

MarksPhysics[i] = scanner.nextInt();

System.out.print ("Enter marks in chemistry: ");

MarksChemistry[i] = scanner.nextInt();

}

}

*public* void calculateTotal ()

{

*// System.out.println ("\nTotal marks for students ");*

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

MarksTotal[i] = (MarksMaths[i] + MarksPhysics[i] + MarksChemistry[i]);

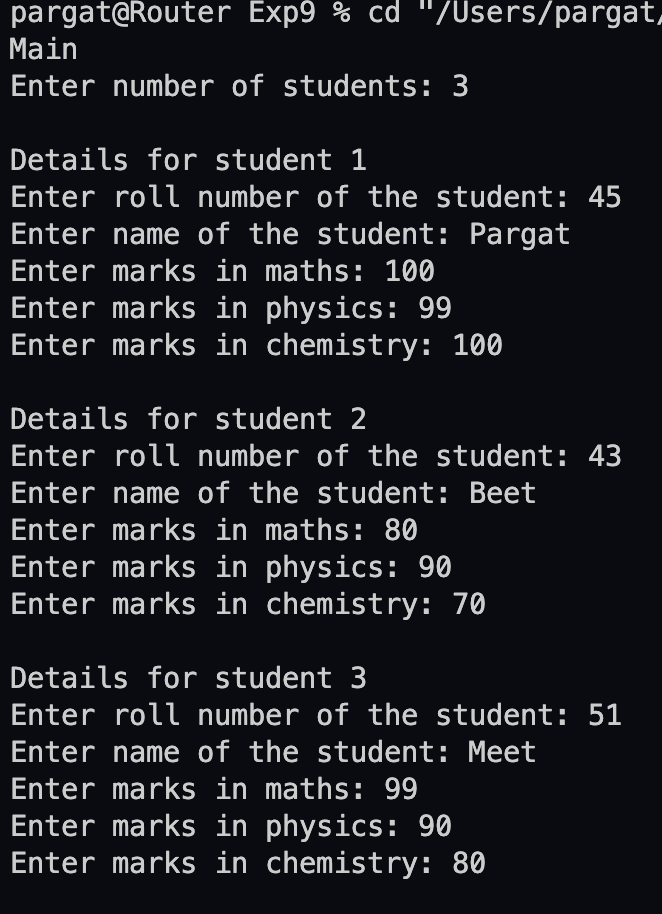
*// System.out.print (MarksTotal[i] + " ");*

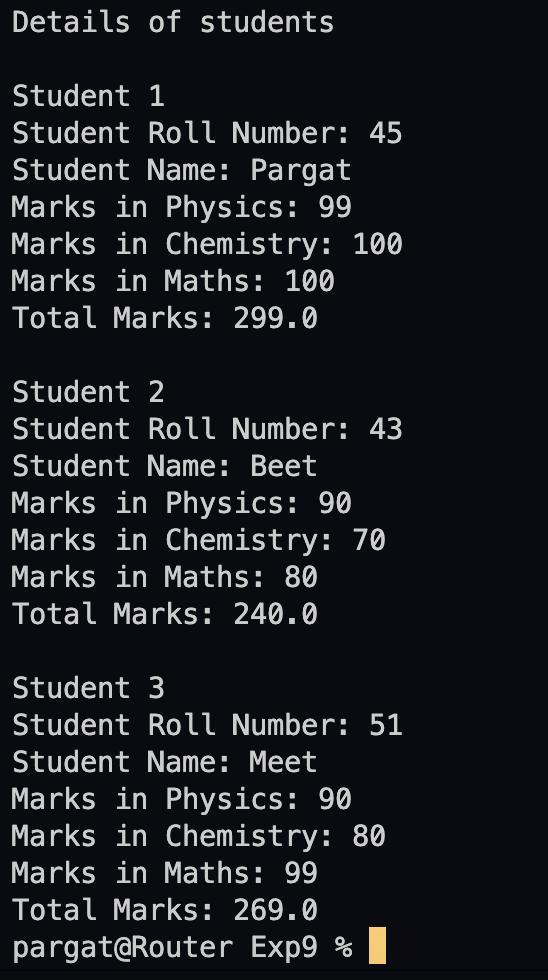
}

}

}

**Output:**





**Conclusion:**

The concept of packages is understood and successfully implemented.

**Date: \_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions**

Q.1   What are Java Packages? What's the significance of packages?

Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations ) providing access protection and namespace management.

Since the package creates a new namespace there won't be any name conflicts with names in other packages. Using packages, it is easier to provide access control and it is also easier to locate the related classes.

 Q.2 Does Importing a package imports its sub-packages as well in Java?

In java, when a package is imported, its sub-packages aren’t imported and the developer needs to import them separately if required.

For example, if a developer imports a package university.\*, all classes in the package named university are loaded but no classes from the sub-package are loaded. To load the classes from its sub-package ( say department), the developer has to import it explicitly as follows: Import university.department.\*