

| **TITLE : Perfect Number** |
| --- |

**AIM:** Define a class Perfect which accepts the range of numbers from the user. Create a static function check\_per , which checks if the number is a perfect number or not and sends the result back to the main function which counts and displays the perfect numbers within that range.

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:**

**Algorithm:**

**Implementation details:**

**Output:**

**Conclusion:**

**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.

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:**

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

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



| **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:**

**Algorithm:**

**Implementation details:**

**Output:**

**Conclusion:**

**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.

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

1+½+⅓+....+1/n



| **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:**

**Algorithm:**

**Implementation details:**

**Output:**

**Conclusion:**

**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.**

**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:**

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



| **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:**

**Algorithm:**

**Implementation details:**

**Output:**

**Conclusion:**

**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:**

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



| **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:**

**Algorithm:**

**Implementation details:**

**Output:**

**Conclusion:**

**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:

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



| **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:**

**Identify Attributes for each class:**

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

**Class Diagram:**

**Algorithm:**

**Implementation details**: (Class Diagram and Code)

**Output:**

**Conclusion**

**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?

b). What are the instance variables?

c.)What are local variables?

**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);

    }

}



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

**AIM:**

Create a user defined exception subclass NumberException with necessary constructor and overridden toString method. Write a program which accepts a number from the user. It throws an object of the NumberException class if the number contains digit 3 otherwise it displays the appropriate message. 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:**

**Implementation details :**

**Output:**

**Conclusion:**

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

**Post Lab Descriptive Questions**

1. Compare throw and throws.
2. Explain how to create a user defined exception and explicitly throw an exception in a program with a simple example.
3. 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?
* If the exception is not caught, will statement4 be executed?
* If the exception is caught in the catch block, will statement4 be executed?
* If the exception is passed to the caller, will the statement4 be executed?

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



| **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:**

**Implementation details:**

**Output:**

**Conclusion:**

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

**Post Lab Descriptive Questions**

1. What do you mean by multithreading?

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

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



| **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:**

**Algorithm:**

**Implementation details:**

**Output:**

**Conclusion:**

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

**Post Lab Descriptive Questions**

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

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