20/8/24

Unit-3

Exception:-

Error:-

An error is a mistake and then given wrong result on the output. A software engineer may also commit several errors while designing the project. or developing the code. These errors are also called "bugs" and then processing of removing those bugs is called "debugging."

Types of Errors:-

There are basically 2 types of errors

1. Compile time Errors 2. Run time Errors

1. Compile time Errors:-

These errors called syntatical errors found in the code. due to which a programe fails to compile. In compile time errors, detect by compiler and those errors handeled by programers.

Program:-

```
class error
{
  public static void main (String args[])
  {
    int a=4, b=2;
    int c = a/b  // missing semicolon (;) *//
    System.out.println ("Result =" + c);
  }
}
```

Output:-

Error: missing semicolon (;)

Run time Error :-

These errors found in Run time environment, that means run time errors detect by JVM (Java Virtual machine) (or) (Java interpreter) and those errors handled by exception handlers. The Run time errors is also known as exception

Program :-

```
class error
{
  public static void main (String args[])
  {
    int a=4, b=0;
    int c=a/b;
  System.out.println ("Result =" +c);
  }
}
```

output :-

Arithmetic error; Can't division by zero.

Predefine exception :-

Exception type	Cause of Exception:
Arithmetic Exception	Caused by mathematical error
ArrayIndexOutOfBoundsException	Caused by invalid Index value
IOException	Caused by input output failures
numberformat Exception	Caused by numberformat problem

Security Exception	Caused by security issues
File Not found Exception	Caused by file cannot found
Null pointer Exception	Caused by Null pointing problem.

Exception Handlers :-

An exception is a condition caused by runtime error in a program. whenever the java interpreter finds the runtime error it creates an exception object. These Exception objects handles Exception handlers.

The exception handling code basically consist of 2 segments, one to detect errors and throw exception and second is to catch the exception and take appropriate actions.

Types of exception Handlers :-

There are 3 types of exception handlers :-

1. Try block
2. Catch block
3. finally block

① Try blocks :-

The try is a keyword in java it is used to perform detect an errors and throws an exception to the catch blocks.

Syntax :-

```
try
{
  Caused error statement
}
```

② Catch blocks :-

The catch is a keyword in java. It is used to perform catch the error object (exception) & handles that exception thrown by the try blocks. If there are more than 1 catch blocks are defined for a single try block then appropriate catch block will receive the exception thrown by the try blocks.

Syntax :-

```
catch (Error type Exception Error object 1)
{
  Exception Handling
}
- - - - -
- - - - -
catch (error type Exception Error object n)
{
   Exception Handling
}
```

③ Finally blocks :-

The finally is a keyword. It is used to perform reallocate the program. It Specify a blocks of statement to execute when none of the catch blocks call the exception. It is optional in exception handlers. A try block can have only one finally block.

Syntax:-

```
finally
    {
      Re allocate Statements.
    }
```

Program:-

```
class myexception
{ public static void main (String args[])
  {
    int a = Integer.parseInt (args[0]);
    int b = Integer.parseInt (args[1]);
   try
   { int c = a/b;
     System.out.println ("Result =" +c);
   }
 Catch (Arithmetic Exception e)
   {
     System.out.println (a+" can not division by zero (0),
                         please change it !" +e);
   }

finally
    {
      System.out.println ("Exit & end of the program");
    }
  }
}
```

multiple catch:-

Eg:-

```
class myExcept
{
  public static void main(String args[])
  {
    int a=Integer.parseInt(args[0]);
    int b=Integer.parseInt(args[1]);
    int c[]={30,40,25};
  try
  {  int r=a/b;
System.out.println("The result="+r);
System.out.println("An array value="+c[]);
}
 catch (ArithmeticException e)
 {
  System.out.println("cannot divisible by zero...");
 }
 catch (ArrayIndexoutofBoundsException )
 {
   System.out.println("An array index out of bound...");
 }
finally
{
  System.out.println("the program is completed");
  }
}
}
```

Packages :-

```
import java.util.*;
import java.lang.*;
import java.io.*;
class Example
{
  public static void main (String args[])
  {
    int x, y;
    Scanner s= new Scanner (System.in);
    System.out.println ("Enter x, y values");
    x= s.nextInt();
    y= s.nextInt();
  System.out.println ("max of two values = "+ math.max(x,y));
System.out.println ("min of two values = "+ math.min (x,y));
System.out.println ("square root of x= "+ math.sqrt (x));
System.out.println ("x power of y= "+ math.pow(x,y));
System.out.println ("Absolute value of x= "+ math.abs (-x));
  }
}
```

User defined Package :-

Text editor

Eg :-

```
package mypackage;
public class Addition
{
  int a,b;
  Public void compute (int a, int b)
  {
    this.a=a;
    this.b=b;
  System.out.println ("the Addition of two values= "+(a+b));
  }
}
```

class name & filename same
Save - Addition.java
compile: Javac -d . Addition.java

Text Editor-2

```
import mypackage.Addition
class ex
{
  public static void main (String args[]).
  {
    Addition obj = new Addition ();
    obj. compute (10,5);
  }
}
```

Save → Ex.Java
Compile → Javac Ex.Java
Run → Java ex.

Write a program in Java to compute Arithmatic calculations using packages:

A) Package Arithmetic;

Text Editor-1

```
Public class Addition
{
  int a,b;
  public void compute (int a, int b)
  {
    this.a=a;
    this.b=b;
    System.out.println ("Add of two val = "+(a+b));
  }
}
```

Text Editor-2

```
package Arithmetic;
public class subtraction
{
  int a,b;
  public void compute (int a, int b)
  {
    this.a=a;
    this.b=b;
    System.out.println ("subs of two val. = "+(a-b));
  }
}
```

Text Editor-3

```
package Arithmatic;
 public class multiplication
{
  int a,b;
  public void compute (int a, int b)
  {
    this.a=a;
    this.b=b;
```

```
System.out.println("mul of two values = "+(a*b));
}
}
```

Text Editor 4

```
package Arithmetic;
public class Division
{ float a,b;
  public void compute (float a,; float b)
  {
    this.a=a;
    this.b=b;
  System.out.println(" Division = "+ (a/b));
  }
}

import Arithmatic.Addition;
import Arithmatic.subtraction;
import Arithmatic.multiplication;
import Arithmatic.Division;
Class Ex
{
  public static void main (String args[])
{
 Addition obj1 = new Addition();
 obj1.compute (10,5);
 subtraction obj2= new subtraction ();
 obj2.compute (10,5);
 multiplication obj3= new multiplication();
 obj3.compute (10,5);
 Division obj4 = new Division ();
  obj4.compute (10,5);
  }
}
```

Package files:-

Save → class name . java.

Compile → Javac -d . classname . java.

importing file :-

Save → class name . java.

Compile → javac class name . java.

Run → java class.name.

Packages :-

A java package is a group of similar type of classes, interfaces, sub packages. A java package is a sub-directory. Packages in java two categories. Package is a keyword in Java

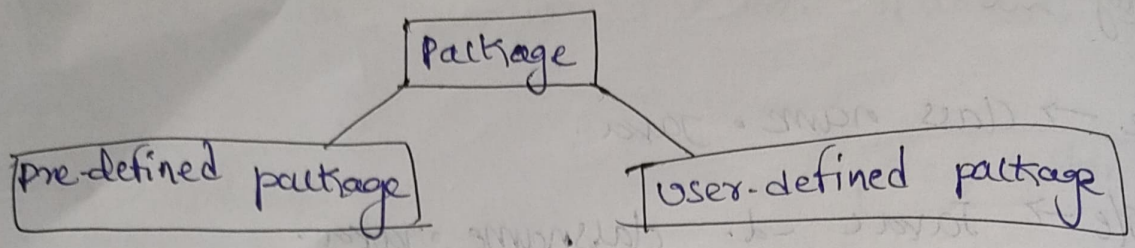
1. Built in package (predefined package).
2. User defined package.

Advantages of Packages :-

1. Java package is used to categories the classes and interfaces
2. It can be easily managed
3. Java package provides access protection.
4. Java package removes "name collision"

Types of Packages :-

There are two types of packages



Pre-defined packages :-

pre-defined package consists large no. of classes, which are part of Java API (Application program interface).

Sum of the commonly used predefined packages

User-defined Packages :-

Package name	Usage.
import java.util.*	Contains utility classes. such as scanner, Date, & time and linked list operation.
import java.io.*	contains classes for input & output operations.
import java.lang.*	Contains language support classes such as primitive data types, mathematical operations.
import java.awt.*	Contains classes for implimenting the components such as buttons, check box, meanus and so on....

import java.Applet.*	Contains classes for creating Applets
import java.net.*	contain classes for supporting networking operations.

Program :-

```
import java.util.*;
import java.lang.*;
import java.io.*;
class example
{
public static void main (String args[])
{
  int x,y;
  Scanner sc = new Scanner (System.in)
  System.out.println ("Enter x,y values");
  x = sc.nextInt ();
  y = sc.nextInt ();
 System.out.println ("max of two values = "+ (Math.max(x,y));
System.out.println ("square root of x=" + math.sqr(x));
}
}
```

User-defined package:-

These are designed or created by the developer to categories classes and packages. They are much similar to the built in packages. It can be imported into the other classes and used the same as we use predefined packages.

To create a package, we are used to the package keyword with package name.

Syntax:-

```
package <package_name>;
```

Ex:-

```
package mypackage.
```

Steps to Creating user-defined package:-

Step 1: Creating a package in java class. Just write a package by its following its package name.

Step 2: Both class name and java file name must be same.

Program:-

Text-Ed-1

```
package mypackage;
public class Addition
{
    int a,b;
    public void compute (int a, int b)
    {
        System.out.println ("Addition =" +(a+b));
    }
}
```

① class name & file name same
Save – Addition.Java.
Compile – Javac -d Addition.Java

```
import mypackage.Addition.
class ex.
{ public static void main (String args[])
  {
  Addition obj = new Addition();
  obj.compute (10,5);
  }
}
```

Text Ed-2

Save → Ex.Java
Compile – Javac ex.java
Run – Java ex

In the above program, you have saved as class both class name and file name must be same.

Save → addition.Java

Compile → Java -d . addition.Java

Step 3: After compilation, it generates the class file with package and then import the ~~key~~ package to the another class, here "import" is a key word.

Example:-

```
import mypackage.Addition.
class ex
  {
  public static void main (String args[])
  {
  Addition obj = new Addition();
  obj.compute (10,5);
  }
}
```

Save → Ex.Java.
Compile – Javac ex.java
Run – Java ex

Notepad-1

```
Package myinterpackage;
Public interface myinterface.
{
final static double Pi=3.14;
Public double Rectangle (double l, double b);
Public double Circle (double r);
}
```

Save → my-interface.Java
Compile → Javac -d . myinterface.Java.

Notepad-2

```
import myinterpackage.myinterface;
Class Area implements myinterface.
{
  Public double Rectangle (double l, double b)
  {
     return (l*b);
  }
  Public double Circle (double r)
  {
     return (pi*r*r);
  }
  Public static void main (String args[])
  {
   Area obj1 = new Area ();
System.out.println ("Area of Rectangle = "+ obj1.Rectangle
                                    (4.0, 2.0));
```

```
System.out.println("Area of circle = " + obj1.circle(2.5f));
    }
}
```

Save → myinterface.java → Area.java

Compile → javac myinterface.java → javac Area.java

Run → java myinterface → java Area