Packages

- A package is a container of classes and interfaces.
- A package represents a directory that contains related group of classes and interfaces.
- For example, when we write statements like: import java.io.*; Here we are importing classes of java.io package. Here, java is a directory name and io is another sub directory within it. The '*' represents all the classes and interfaces of that io sub directory.
- We can create our own packages called user-defined packages or extend the available packages.
- User-defined packages can also be imported into other classes and used exactly in the same way as the Built-in packages.
- Packages provide reusability.

General form for creating a package:

package packagename;

e.g.: package pack; package lubak;

- The first statement in the program must be package statement while creating a package.
- While creating a package except instance variables, declare all the members and the class itself as public then only the public members are available outside the package to other programs.

Program 1: Write a program to create a package pack with Addition class.

```
//creating a package
package pack;
public class Addition
{
private double d1,d2;
public Addition(double a,double b)
{
d1 = a;
d2 = b;
}
public void sum()
{
 System.out.println ("Sum of two given numbers is : " + (d1+d2) );
}
}
```

Compiling the above program:

Compiling the above program:

➔The –d option tells the Java compiler to create a separate directory and place the .class file in that directory (package).

➔The (.) dot after –d indicates that the package should be created in the current directory. So, output package pack with Addition class is ready.

Program 2: Write a program to use the Addition class of package pack.

```
//Using the package pack
import pack.Addition;
```

```
class Use

{
public static void main(String args[])
{
Addition ob1 = new Addition(10,20);
ob1.sum ();
}
}
```
Output:……

Program 3: Write a program to add one more class Subtraction to the same package pack.

```
//Adding one more class to package pack:
package pack;
public class Subtraction
{
private double d1,d2;
public Subtraction(double a, double b)
{
d1 = a;
d2 = b;
}
public void difference()
{
 System.out.println ("dnce of two given numbers is : " + (d1 - d2) );
}
}
```
Compiling the above program:……..

Compiling the above program:

Program 4: Write a program to access all the classes in the package pack.

```
//To import all the classes and interfaces in a class using import pack.*;
import pack.*;
class Use
{
public static void main(String args[])
{
Addition ob1 = new Addition(10.5,20.6);
ob1.sum();
Subtraction ob2 = new Subtraction(30.2,40.11);
ob2.difference();
}
}
```
➔***In this case, please be sure that any of the Addition.java and Subtraction.java programs will not exist in the current directory***. Delete them from the current directory as they cause confusion for the Java compiler. The compiler looks for byte code in Addition.java and Subtraction.java files and there it gets no byte code and hence it flags some errors.

Output:……

Exceptions

- ❖ An error in a program is called bug. Removing errors from program is called debugging.
- ❖ There are basically three types of errors in the Java program:

1) **Compile time errors:** Errors which occur due to syntax or format is called compile time errors. These errors are detected by java compiler at compilation time. Desk checking is solution for compile-time errors.

2) **Runtime errors:** These are the errors that represent computer inefficiency. Insufficient memory to store data or inability of the microprocessor to execute some statement is examples to runtime errors. Runtime errors are detected by JVM at runtime.

3) **Logical errors:** These are the errors that occur due to bad logic in the program. These errors are rectified by comparing the outputs of the program manually.

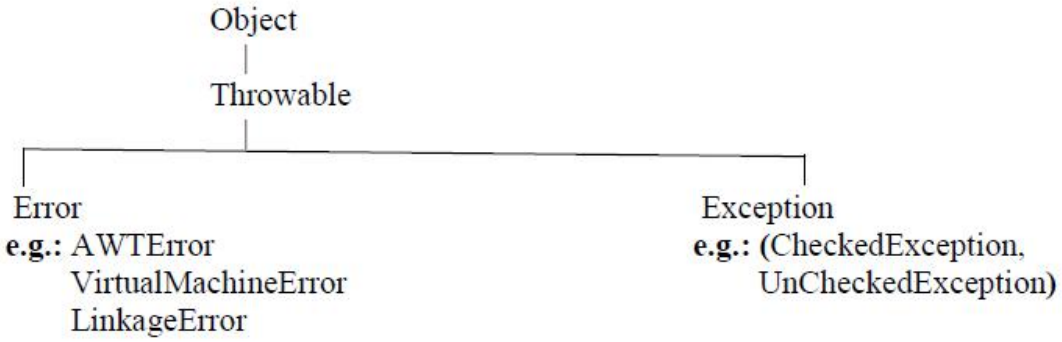
Exception: An abnormal event in a program is called Exception.

➔Exception may occur at compile time or at runtime.

A) Exceptions which occur at compile time are called **Checked exceptions**.
e.g.: ClassNotFoundException, NoSuchMethodException, NoSuchFieldException etc.

B) Exceptions which occur at run time are called **Unchecked exceptions**.
eg: ArrayIndexOutOfBoundsException, ArithmeticException, NumberFormatException etc.

➔**Exception Handling:** Exceptions are represented as classes in java.



An exception can be handled by the programmer where as an error cannot be handled by the programmer. When there is an exception the programmer should do the following tasks:

- If the programmer suspects any exception in program statements, he should write them inside try block.

```
try
{
        statements;
}
```

- When there is an exception in try block JVM will not terminate the program abnormally. JVM stores exception details in an exception stack and then JVM jumps into catch block. The programmer should display exception details and any message to the user in catch block.

```
catch ( ExceptionClass obj)
{       statements;
}
```

- Programmer should close all the files and databases by writing them inside finally block. Finally block is executed whether there is an exception or not.

```
finally
{               statements;
}
```

- Performing above tasks is called Exception Handling.

Example 1:**Program 1:** Write a program which tells the use of try, catch and finally block.

```
// Exception example
class ExceptionExample
{
public static void main(String args[])
{
try
{
System.out.println ("open files");
int n=args.length;
System.out.println ("n="+n);
int a=45/n;
System.out.println ("a="+a);
int b[]={10,19,12,13};
b[50]=100;
}
catch (ArithmeticException ae)
{
System.out.println ("ae");
System.out.println ("plz type data while executing the program");
}

catch (ArrayIndexOutOfBoundsException aie)
```

```
{
System.out.println ("aie");
System.out.println ("please see that array index is not within the range");
}

finally
{
 System.out.println ("close files");
}
}
}
```

Output :

- Even though multiple exceptions are found in the program, only one exception is raised at a time.
- We can handle multiple exceptions by writing multiple catch blocks.
- A single try block can be followed by several catch blocks.
- Catch block does not always exit without a try, but a try block exit without a catch block.
- Finally block is always executed whether there is an exception or not.

- **throws Clause: throws clause is useful to escape from handling an exception. throws clause is useful to throw out any exception without handling it.**

Program 2: Write a program which shows the use of throws clause.

```
// not handling the exception
import java.io.*;
class Sample
{
void accept( )throws IOException
{
BufferedReader br=new BufferedReader (new InputStreamReader(System.in));
System.out.print ("enter ur name: ");
String name=br.readLine ( );
System.out.println ("Hai "+name);
}
}
class ExceptionNotHandle
{ public static void main (String args[])throws IOException
{ Sample s=new Sample ( );
s.accept ( );
}}
```
