Dt : 7/9/2023

1.Error class:

=>The disturbance which is occured from the environment is known as "error"

=>"java.lang.Error" class is the SuperClass or ParentClass of all the errors raised from the environment.

=>There is no separate process to handle errors.

****imp***

2.Exception class:

=>The disturbance occured from the application is known as Exception.

=>"java.lang.Exception" is the SuperClass or ParentClass of all the exceptions raised from the application.

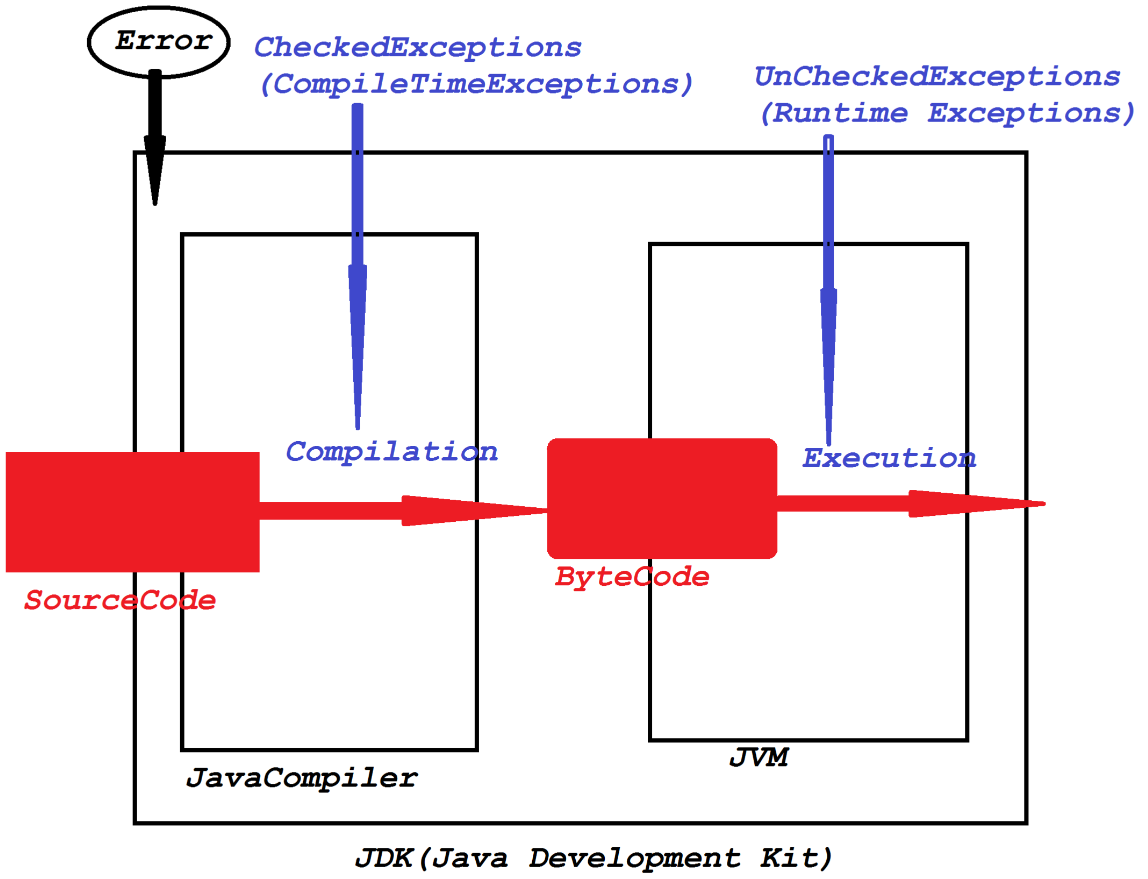
--

Types of Exceptions:

=>Exceptions are categorized into two types:

1.UnChecked Exceptions

2.Checked Exceptions



1.UnChecked Exceptions:

=>The exceptions which are not identified by the Compiler at compilation stage will be raised at execution stage are known as UnChecked Exceptions or Runtime Exceptions.

=>These UnChecked Exceptions are categorized into two types:

(a)Pre-defined UnChecked Exceptions

(b)User defined UnChecked Exceptions

(a)Pre-defined UnChecked Exceptions:

=>The UnChecked Exceptions which are already defined and available from JavaLib are known as Pre-defined UnChecked Exceptions.

Ex:

java.lang.NumberFormatException

java.util.InputMismatchException

...

(b)User defined UnChecked Exceptions:

=>The UnChecked Exceptions which are defined and raised by the programmer are known as User defined UnChecked Exceptions.

=>We use the following steps to define and raise User defined UnChecked Exceptions:

step-1 : The user defined class must be extended from pre-defined class "java.lang.Exception"

step-2 : The user defined class must be declared with parameterized constructor with string as parameter

step-3 : This parameterized constructor must pass the msg to the PClass using "super()".

step-4 : The exception-statements must be declared under try-block

step-5 : Define Exception-condition to raise the exception

step-6 : when the Exception condition is true,then create object for

User defined class and while object creation pass

exception-msg as parameter

step-7 : we use "throw" keyword to throw object reference onto catch

block

step-8 : The exception-msg must be displayed from catch-block

Ex:

ProjectName : App_ExceptionHandlingProcess

packages,

maccess : DemoException1.java(MainClass)

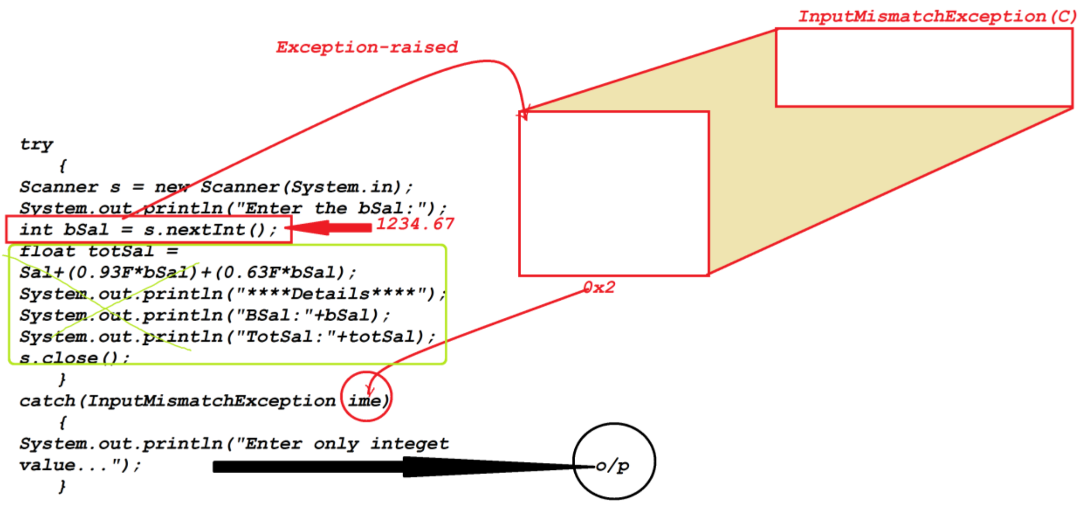
```
package maccess;
import java.util.*;
public class DemoException1 extends Exception
{
     public DemoException1(String msg)
     {
          super(msg);
     }
     public static void main(String[] args)
     {
          Scanner s = new Scanner(System.in);
          try
          {
                    System.out.println("Enter the bSal:");
                    int bSal = s.nextInt();
                    //Exception for NonInteger input
                    if(bSal<12000)//Exception Condition
                    {
                     DemoException1 de = new DemoException1
                              ("Invalid bSal...");
                     throw de;
                    }
                    float totSal =
bSal+(0.93F*bSal)+(0.63F*bSal);
                    System.out.println("****Details****");
                    System.out.println("BSal:"+bSal);
                    System.out.println("TotSal:"+totSal);
          }
          catch(InputMismatchException ime)
          {
               System.out.println("Enter only integet
value...");
          }
          catch(DemoException1 de)
          {
               System.out.println(de.getMessage());
          }
          finally
          {
               s.close();
          }

     }
}
```

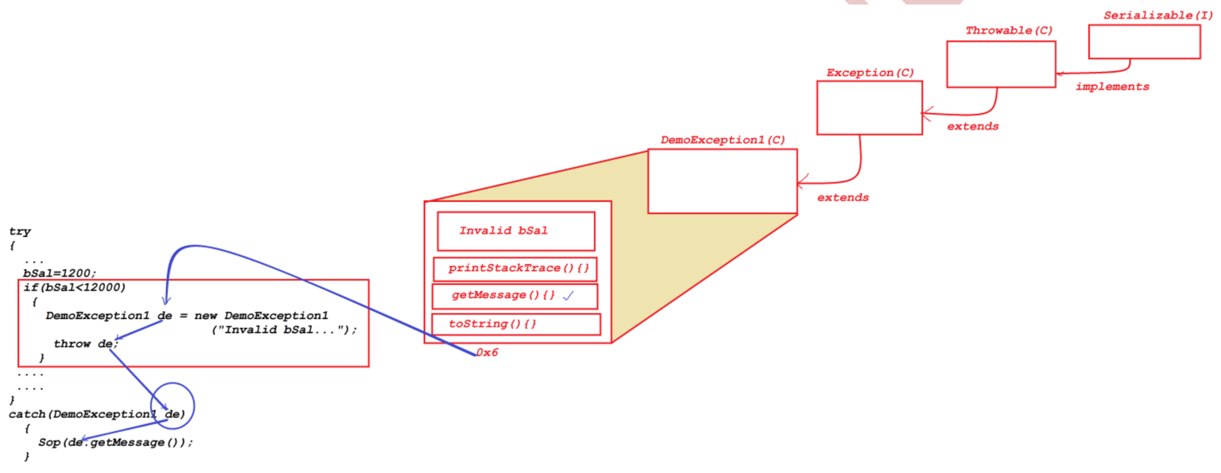
o/p:

```
Enter the bSal:
12000
****Details****
BSal:12000
TotSal:30720.0
```

Diagrams:



```
try
    {
Scanner s = new Scanner(System.in);
System.out.println("Enter the bSal:");
int bSal = s.nextInt();
~~float totSal =~~
~~Sal+(0.93F*bSal)+(0.63F*bSal);~~
~~System.out.println("****Details****");~~
~~System.out.println("BSal:"+bSal);~~
~~System.out.println("TotSal:"+totSal);~~
~~s.close();~~
    }
catch(InputMismatchException ime)
    {
System.out.println("Enter only integet
value...");
    }
```



```
try
{
  ...
  bSal=1200;
  if(bSal<12000)
   {
     DemoException1 de = new DemoException1
                              ("Invalid bSal...");
      throw de;
    }
 ....
 ....
}
catch(DemoException1 de)
   {
     Sop(de.getMessage());
   }
```

==