UNIT 4 - Exception Handling

Information regarding Exception :-

- Dictionary meaning of the exception is abnormal termination.
- An exception is a problem occurred during execution time of the program.
- An unwanted unexpected event that disturbs normal flow of execution called exception.
- Exception is nothing but a object.
- Exception is a class present in java.lang package.
- All the exceptions are nothing but objects of classes.
- Whenever user is entered invalid data then Exception is occur.
- A file that needs to be opened can't found then Exception is occurred.
- Exception is occurred when the network has disconnected at the middle of the communication.

Types of Exceptions:-

As per sun micro systems standards The Exceptions are divided into three types

1) Checked Exception
2) Unchecked Exception
3) Error

1) Checked Exception:-

The Exceptions which are checked by the compiler at compilation time for the proper execution of the program at runtime is called Checked Exceptions.

Ex:- IOException, SQLException etc…….

Some of the checked Exceptions in the java language

Exception	Description
ClassNotFoundException	If the loaded class is not available
IllegalAccessException	Access to a class is denied.
InstantiationException	Attempt to create an object of an abstract class or interface.
InterruptedException	One thread has been interrupted by another thread.
NoSuchFieldException	A requested field does not exist.
NoSuchMethodException	If the requested method is not available

2) Unchecked Exception:-

The exceptions which are not checked by the compiler at compilation time is called unchecked Exception. These checking down at run time only.

Ex:- ArithmeticException, NullPointerException, etc…….

Some of the unchecked exceptions in the java language:

Exception	Description
ArithmeticException	Arithmetic error, such as divide-by-zero.
ArrayIndexOutOfBoundsException	Array index is out-of-bounds.(out of range)
InputMismatchException	If we are giving input is not matched for storing input
ClassCastException	If the conversion is Invalid.

IllegalArgumentException	Illegal argument used to invoke a method.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	Array created with a negative size.
NullPointerException	Invalid use of a null reference.
NumberFormatException	Invalid conversion of a string to a numeric format.
StringIndexOutOfBoundsException	Attempt to index outside the bounds of a string.

3) Error:-
Errors are caused by lack of system resources. These are non recoverable.
Ex:- StackOverFlowError, OutOfMemoryError, AssertionError etc.....

The Exception whether it is checked or unchecked the Exceptions are occurred at runtime.

Difference between Exception and Error:-
Exception:- An exception is unwanted unexpected event these are caused by the programmer mistake. Exceptions are recoverable.
Ex:- IOException, SQLExcetion, RuntimeExecption etc.....

Error:- Errors are caused by lack of system resources. These are non recoverable.
Ex:- StackOverFlowError, AssertionError etc.....

Exception Handling :
Exceptions can be handled in two ways 1.By using try-catch blocks 2. By using throws keyword.

Exception handling by using try–catch block:- 1) In java language we are handling the exceptions By using try and catch blocks. try block contains risky code of the program and catch block contains handling code of the program. 2) Catch block code is a alternative code for Exceptional code. If the exception is raised the alternative code is executed fine then rest of the code is executed normally.

Syntax:-
```
try {
    Riskey Code
  }
 Catch(ExceptionName reference_variable) {
    Code to run if an exception is raised
 }
```

Before try and catch:-The program goes to abnormal termination.
```
class Test {
    public static void main(String[] args) {
    System.out.println("durga");
    System.out.println("software");
    System.out.println(10/0);
    System.out.println("solutions");
   } }
```

Output:-

Durga
Software
Exception in Thread “main” :java.lang.ArithmeticException: / by zero

Note:- if we are not using try-catch it is always abnormal termination if an Exception raised.

After try catch:-

If we are taking try-catch the program goes to normal termination. Because the risky code we are taking inside the try block and handling code we are taking inside the catch block. 2) If the exception is raised in the try block the corresponding catch block is executed. 3) If the corresponding catch block is not there program goes to abnormal termination.

```
class Test {
    public static void main(String[] args) {
       System.out.println("durga");
       System.out.println("software");
       try {
              System.out.println(10/0);
         }
       catch (ArithmeticException e) {
              System.out.println("you are getting AE "+e);
         }
       System.out.println("solutions");
       } }
```

Output:-

Durga
Software
You are getting AE: java.lang.ArithmeticException: / by zero
Solutions.

Points to be noted when using try-catch :

1. Exception raised in try block the JVM will search for corresponding catch block if the catch block is matched, corresponding catch block will be executed and rest of the code is executed normally. If the catch block is not matched the program is terminated abnormally.
2. If there is no exception in try block the catch blocks won’t be executed.
3. Independent try blocks are not allowed in java language with out catch() or Finally.(compilation error)
4. In between try and catch independent statements are not allowed. If we are providing independent statements the compiler will raise compilation error.
5. Once the control is out of the try block the control never return back to try block again.
6. Multiple catch blocks can be used with single try blocks, to handle multiple exceptions possible.
7. By using root class (Exception) we are able to hold any type of exceptions.
8. In java class if we are declaring multiple catch blocks at that situation the catch block order should be child to parent shouldn’t be parent to the child.
9. The exception raised in catch block it is always abnormal termination.

Possibilities of try-catch blocks:-

1**) single time try-catch:-**

```
try {
}
catch () {
}
```

2) **multiple times try-catch:-**

```
try {
}
catch () {
}
try {
}
catch () {
}
```

3) **try with multiple catchs:-**

```
try {
}
catch () {
}
catch () {
}
```

4) nested try-catch:-

```
try {
  try {
  }
  catch () {
  }
}
catch () {
}
```

5) **catch with try-catch:-**

```
try {
}
catch () {
try {
}
catch () {
}
}
```

6) **Nesting in both try-catch**

```
try {
  try {
  }
  catch () {
  }
}
catch () {
  try {
  }
  catch () {
  }
}
```

Finally Block:

1) Finally is a block it is always executed irrespective of try and catch.
2) Finally contains clean-up code.
3) It is not possible to write finally alone . we must take try-catch-finally otherwise take the tryfinally these two are the possibilities. If we are taking any other we are getting compilation error saying finally without try block .

Syntax:-

```
try {
 risky code;
}
catch (Exception obj) {
   handling code;
}
 finally {
   free code;
 }
```

Ex :-
Exception raised in try block and corresponding catch block is matched then rest of the code is executed normally.

```
class Test {
 public static void main(String[] args)
 {
 try {
 System.out.println("durga");
System.out.println(10/0);
}
 catch (ArithmeticException ae) {
   System.out.println("u r getting ae:"+ae);
 }
 finally {
 System.out.println("finally block is always executed");
 }
 System.out.println("rest of the code");
 }
 }
```

Output:-

```
Durga
U r getting ae:ArithmeticException : /by zero
Finally block is always executed
```

Points to be noted :

1) Exception raised in try block and corresponding catch block is matched then rest of the code is executed normally along with Finally.
2) Exception raised in try block and corresponding catch block is not matched then after executing finally the program terminate abnormally.
3) The only one situation the finally block is wont be executed is, in the program if we use System.exit(0) then JVM shutdown hence the rest of the code won't be executed .

Throw Keyword:

1. The main purpose of the throw keyword is to creation of Exception object explicitly either for predefined or user defined.

2. Throw keyword works like a try block. The difference is try block is automatically find the situation and creates a Exception object implicitly. Whereas throw keyword creates a Exception object explicitly.

```
class Test {
  public static void main(String[] args) {
  try {
    System.out.println(10/0);
  }
 catch (ArithmeticException ae) {
   System.out.println("we are getting Exception "+ae);  } } }
```

Output:-
we are getting Exception ArtithmeticException: / by zero

In the above program the main method is responsible to create an exception object. So the main method is creating exception object implicitly. The programmer is not responsible person to create an exception object.

```
Ex:- import java.util.*;
 class Test {
   static void validate(int age) {
     if (age<18){
       throw new ArithmeticException("not elgible for vote");
     }
    else {
       System.out.println("welcome to the voteing");
      }
    }
   public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("please enter your age ");
      int n=s.nextInt();
      validate(n);
      System.out.println(“rest of the code”);
   }
}
```

Throws Keyword:

1) The main purpose of the throws keyword is to bypass the generated exception from present method to caller method.
2) throws is a keyword in Java that is used with signature of a method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

Syntax of Java throws

type method_name(parameters) throws exception_list

exception_list is a comma separated list of all the exceptions which a method might throw.

In a program, if there is a chance of raising an exception then the compiler always warns us about it and compulsorily we should handle that checked exception, Otherwise, we will get compile time error saying **unreported exception XXX must be caught or declared to be thrown**. To prevent this compile time error we can handle the exception in two ways:

1. By using **try catch**
2. By using the **throws** keyword

We can use the throws keyword to delegate the responsibility of exception handling to the caller (It may be a method or JVM) then the caller method is responsible to handle that exception.

Important Points to Remember about throws Keyword

- throws keyword is required only for checked exceptions and usage of the throws keyword for unchecked exceptions is meaningless.
- throws keyword is required only to convince the compiler and usage of the throws keyword does not prevent abnormal termination of the program.
- With the help of the throws keyword, we can provide information to the caller of the method about the exception.

```
public class Test {
static void checkAge(int age) throws ArithmeticException {
if(age<18){
  throw new ArithmeticException("Access Denied – you must be at least 18 years old.");
 }
else {
 System.out.println("Access granted – you are old enough!");
 }

public static void main(String args[]){
checkAge(15);
 }
}
```

Differences between throw and Throws keywords :-

throw	throws
Used to throw an exception for a method	Used to indicate what exception type may be thrown by a method

Syntax :
throw is followed by an object(new type)
`throw new exception_class("error message");`
	Syntax :
	throws is followed by a class
	`return_type method_name() throws exception_class_name{`
	
	`//method code`
	`}`

Using throw keyword we can create both checked and unchecked exceptions.	The throws keyword can be used to propagate checked exceptions only
Cannot throw multiple exceptions	Can declare multiple exceptions
Used inside the method	Used with the method signature

User Defined Exception:

Based on the user requirement user can creates a Exception is called user defined Exception. Ex: InvaliedAgeException, BombBlostException……..etc.

These are the exceptions related to business logic and workflow. It is useful for the application users or the developers to understand the exact problem.

In order to create a custom exception, we need to extend the Exception class that belongs to java.lang package.

To create user defined Exceptions:-

1) To create user defined exception we have to take an user defined class that is a sub class to the RuntimeException(for creation of unchecked Exceptions).
2) To create user defined exception we have to take user defined class that is subclass to the Exception(for creation of checked Exceptions).

User Defined Class for creating Exception :

```
public class InvaliedAgeException extends Exception {
   InvaliedAgeException(String str)  {
     super(str);
   }
}
```

Java Program that uses user defined class for throwing Exception

```
import java.util.*;
class Test {
   static void validate(int age) throws InvaliedAgeException {
      if (age<18)
      {
           throw new InvaliedAgeException("not elgible for vote");
      }
      else
      {
          System.out.println("welcome to the voteing");
      }
   }

  public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("please enter age");
      int age=s.nextInt();
      try{
       validate(age);
      }
           catch(Exception e) {
             System.out.println(e);
      }
           System.out.println("Completed");
 }}
```

Sample output 1 :

please enter age
15
InvaliedAgeException: not elgible for vote
Completed

Sample output 2 :

please enter age
45
welcome to the voteing
Completed