What is exception

Dictionary Meaning: Exception is an abnormal condition.
In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. Exception normally disrupts the normal flow of the application that is why we use exception handling. Let's take a scenario:

Types of Exception

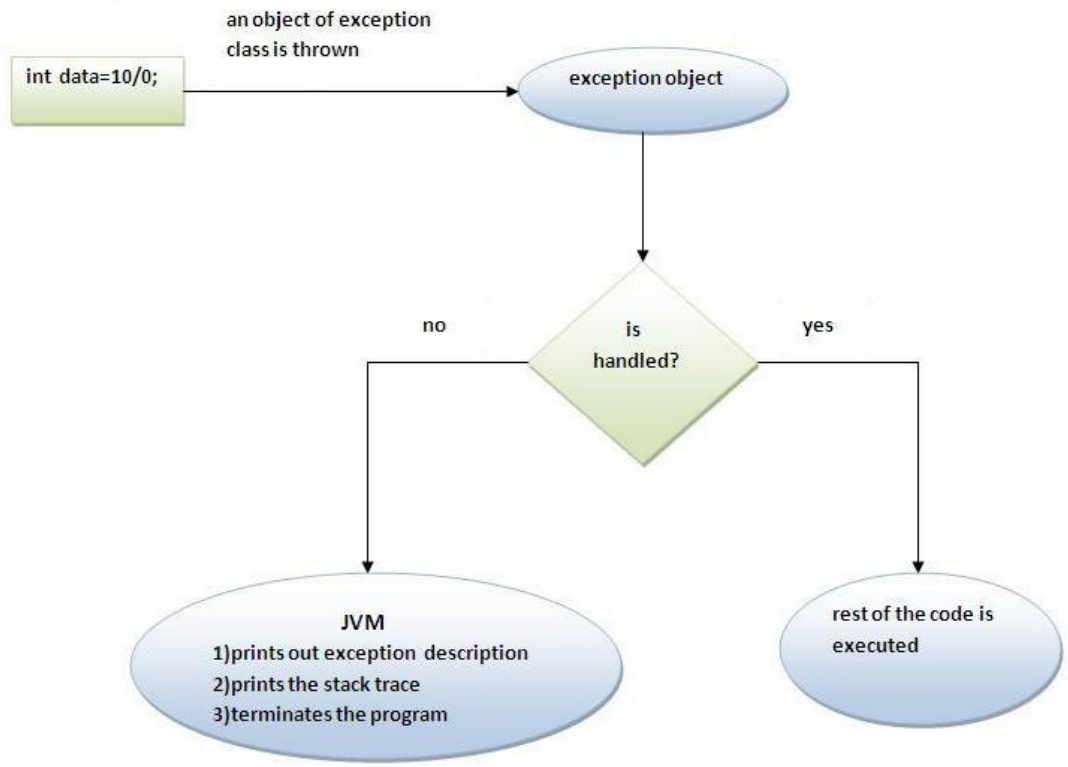
There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions:

1. Checked Exception 2.Unchecked Exception 3.Error

`int a=50/0;//ArithmeticException`
`String s=null;`
`System.out.println(s.length());//NullPointerException String s="abc";`
	`int i=Integer.parseInt(s);//NumberFormatException`
	`int a[]=new int[5];`
	`a[10]=50; //ArrayIndexOutOfBoundsException`

There are 5 keywords used in java exception handling.
Try ,catch ,Finally ,throw ,throws
	`try{ //code that may throw exception`
	`}catch(Exception_class_Name ref){}`
	`----------`
	`try{ //code that may throw exception }finally{}`

Internal working of java try-catch block



Java Multi catch block

```
public class TestMultipleCatchBlock{
  public static void main(String args[]){
   try{
    int a[]=new int[5];       a[5]=30/0;     }
   catch(ArithmeticException e){System.out.println("task1 is completed");
}     catch(ArrayIndexOutOfBoundsException e){System.out.println("task
 2 completed");}     catch(Exception e){System.out.println("common tas
k completed");}
System.out.println("rest of the code...");   }  }
```

```
Output:task1
completed

rest    of    the
code...
```

Rule: At a time only one Exception is occured and at a time only one catch block is executed.

Rule: All catch blocks must be ordered from most specific to most general i.e. catch for ArithmeticException must come before catch for Exception .

Java Nested try block

Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error. In such cases, exception handlers have to be nested.

```
class Excep6{
 public static void main(String args[]){
  try{
    try{       System.out.println("going to divide");
     int b =39/0;
        }catch(ArithmeticException e){System.out.println(e);}
      try{      int a[]=new int[5];      a[5]=4;
    }catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}

    System.out.println("other statement);
  }catch(Exception e){System.out.println("handeled");}
  System.out.println("normal flow..");  }  }
```

Why use java finally

- Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc.

```
class TestFinallyBlock{
  public static void main(String args[]){
  try{
   int data=25/5;
   System.out.println(data);    }
  catch(NullPointerException e){System.out.println(e);}
  finally{System.out.println("finally block is always executed");}
  System.out.println("rest of the code...");    }  }
```

```
Output:5
       finally block is always executed
       rest of the code...
```

Rule: For each try block there can be zero or more catch blocks, but only one finally block.

Note: The finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).

Java throw keyword

The Java throw keyword is used to explicitly throw an exception.

```
public class TestThrow1{
   static void validate(int age){
     if(age<18)
      throw new ArithmeticException("not valid");
     else
      System.out.println("welcome to vote");     }
   public static void main(String args[]){
     validate(13);
     System.out.println("rest of the code...");    }  }
```

Output:

```
Exception in thread main java.lang.ArithmeticException:not valid
```

Java Exception propagation

An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method,If not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack.This is called exception propagation.

```
class TestExceptionPropagation1{
  void m(){     int data=50/0;    }    void n(){
    m();    }
  void p(){     try{      n();     }catch(Exception e){System.out.printl
n("exception handled");}    }
  public static void main(String args[]){
   TestExceptionPropagation1 obj=new TestExceptionPropagation1();
    obj.p();     System.out.println("normal flow...");    } }
```

```
Output:exception
handled
       normal
flow...
```

Java throws keyword

The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

Exception Handling is mainly used to handle the checked exceptions. If there occurs any unchecked exception such as NullPointerException, it is programmers fault that he is not performing check up before the code being used.

Which exception should be declared

Ans) checked exception only, because:

unchecked Exception: under your control so correct your code.

error: beyond your control e.g. you are unable to do anything if there occurs VirtualMachineError or StackOverflowError.

```
import java.io.IOException;
class Testthrows1{
  void m()throws IOException{
    throw new IOException("device error");//checked exception    }
  void n()throws IOException{
    m();    }
  void p(){     try{      n();
   }catch(Exception e){System.out.println("exception handled");}    }
  public static void main(String args[]){
.   Testthrows1 obj=new Testthrows1();
.   obj.p();     System.out.println("normal flow...");    } }
```

```
exception
handled
normal
flow...
```

Difference between throw and throws in Java

There are many differences between throw and throws keywords. A list of differences between throw and throws are given below:

No.	throw	throws
1)	Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
2)	Checked exception cannot be propagated using throw only.	Checked exception can be propagated with throws.
3)	Throw is followed by an instance.	Throws is followed by class.
4)	Throw is used within the method.	Throws is used with the method signature.
5)	You cannot throw multiple exceptions.	You can declare multiple exceptions e.g. public void method()throws IOException,SQLException.

Java throw example

```
void m(){
throw new ArithmeticException("sorry");
}
```

```
void m()throws ArithmeticException{
throw new ArithmeticException("sorry"); }
```

Java throws example

```
void m()throws ArithmeticException{
//method code
}
```

Difference between final, finally and finalize

No	final	finally	finalize
1	Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.	Finally is used to place important code, it will be executed whether exception is handled or not.	Finalize is used to perform clean up processing just before object is garbage collected.
2	Final is a keyword.	Finally is a block.	Finalize is a method.

```
class FinalExample{
public static void main(String[] args){
final int x=100;
x=200;//Compile Time Error
}}
```

```
class FinallyExample{
public static void main(String[] args){
try{
int x=300;
}catch(Exception e){System.out.println(e);}
finally{System.out.println("finally block is executed");} }}
```

```
class FinalizeExample{
public void finalize(){System.out.println("finalize called");}
public static void main(String[] args){
FinalizeExample f1=new FinalizeExample();
FinalizeExample f2=new FinalizeExample();
f1=null; f2=null;
System.gc(); }}
```

ExceptionHandling with MethodOverriding in Java

There are many rules if we talk about methodoverriding with exception handling. The Rules are as follows:

- **If the superclass method does not declare an exception**
  - If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but it can declare unchecked exception.
- **If the superclass method declares an exception**
  - If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

Java Custom Exception

If you are creating your own Exception that is known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need.

```
class InvalidAgeException extends Exception{
 InvalidAgeException(String s){    super(s);   } }
```

```
class TestCustomException1{


  static void validate(int age)throws InvalidAgeException{
    if(age<18)
     throw new InvalidAgeException("not valid");
    else
     System.out.println("welcome to vote");
  }        public static void main(String args[]){
     try{
.     validate(13);
.     }catch(Exception m){System.out.println("Exception occured:
"+m);}
.       System.out.println("rest of the code...");
. } }
```

```
Output:Exception
occured:
InvalidAgeException
:not valid
       rest  of  the
code...
```