**Exception Handling**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Q1. What is an exception?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

  Exception is an event which occurs at program run time and which is responsible for disturbing the normal flow of application called as exception.

**Q2. What is the benefit of exception handling?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Exception handling helps developers to detect the run time error.
2. Exception handling helps us to handle the run time error at program run time means skip the code in which an exception may occur and execute the remaining code in the safe zone.
3. Developers can generate exception warnings in code at program compile time  using checked exceptions.
4. Developer can create own exception and handle it and reuse in project according to requirement of application

**Types of Exception**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Checked Exception:  t**hose exceptions occur at program compile time called checked exceptions.

Checked exceptions, especially used for generating exception warnings may occur in future so better way you can handle that exception at program compile time.

1. **Unchecked Exception:** those exceptions occur at program run time called as unchecked exceptions.

1. **Error :** Error is part of exceptions but not handled by a programmer called as error.

**Q. What is the difference between Error and Exceptions?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Exception may be compile time and run time as well as exception may be handle by developer at compile time or may be at run time but error cannot handle by developer and always occur at program run time and may be occur at program compile time when developer make mistake in syntax

    If we want to work with exceptions in JAVA we have five major keywords.

1. **Try :**  try is blocked in exception handling. We can use it to write code in which exceptions may occur.

Suppose consider we have some logic if developer want logic may be generate exceptions program run time then need to write code  in try block

**Example:** Suppose consider we have program we want to input two values and calculate its division then we have logic c=a/b here if b value is zero then it is consider infinity and system  cannot calculate infinity so here system may be generate run time error so we required to write this logic in try block.

**Note:** When try block generates exception then JVM creates one exception class object in try block according to exception category and hand over to catch for further executions.

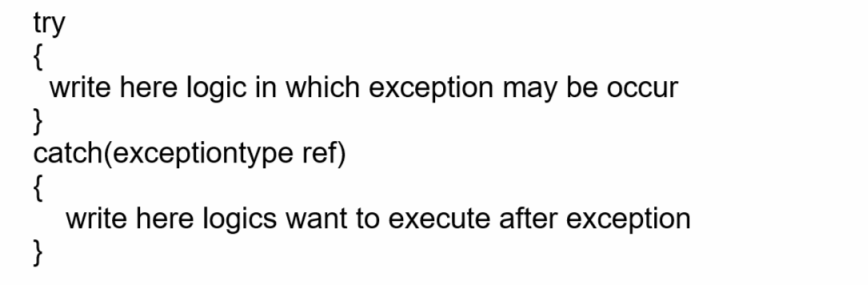
1. **Catch:** catch block always use with try  block and catch block execute when try block generate exception if try block not generate exception then catch not executed

Means we can say catch block specially design for handle the exception and execute the logics after exceptions

Normally we can detect or handle the exception using a catch block.

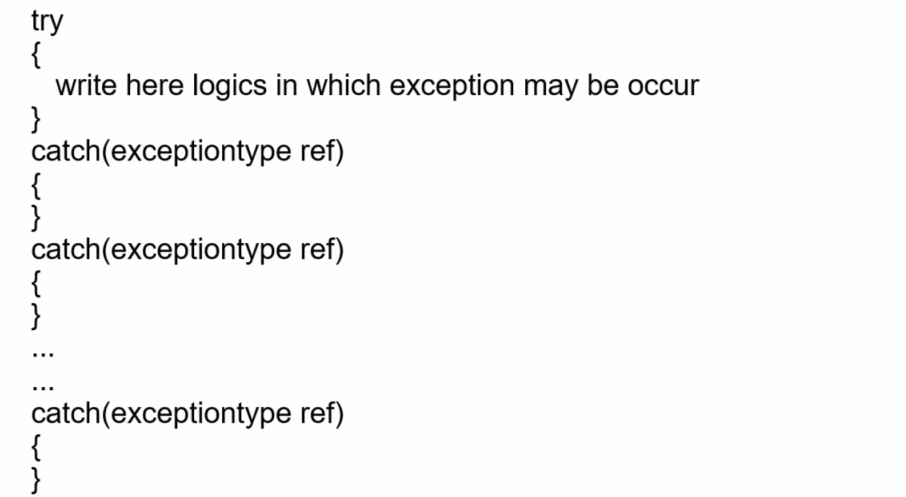
**Syntax:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Or

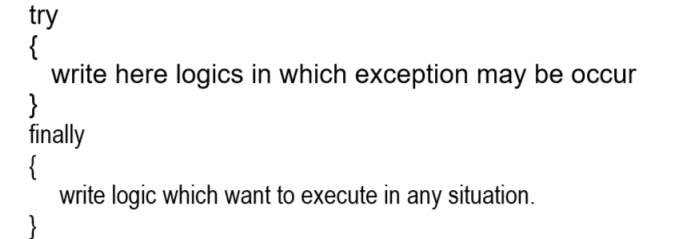
Single try can have more than one catch block.



1. **Finally:** finally is a block in exception handling which always executes if exceptions are generated in code or not.

**Note:** sometimes we have some logic. We are required  to execute any situation if an exception occurs in code or not then we can write that type  of logic in the finally block.

**Example:** database connection close, file connection close etc



**Note:** we can use try catch and finally at the same time.



**Q. Can we write try without catch?**

Yes we can write try without catch using finally block.

**Q. Can we use catch and finally at same time with try block?**

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yes we can use catch and finally at same time with try block but precedence of catch must be before finally means we need to write catch block before finally and after try block shown in above screen shot

**Q. What is the difference between finally , finalize and final?**

**Or**

**Q. What is the difference between final finally and finalize?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Final is non access specifier in java and we can use final keyword with variable , function and class

So when use final keyword with variable then variable mark as constant value means cannot modify later once we initialize it and when we use final keyword with method then method cannot override and when we use final keyword with class then class cannot be inherit in any another class

**Finally :** finally is  block specially we use in exception handling and can use with try block or may be use with catch also and it is used for write logic if exception generate  in code  or not and when we not use catch with try and if program have exception then finally execute before by JVM

**Finalize:** finalize is a method from Object class and which is used for resource cleaning purpose means normally finalize method used in garbage collection process and this method executed automatically when developer call the System.gc() method in garbage collection process.

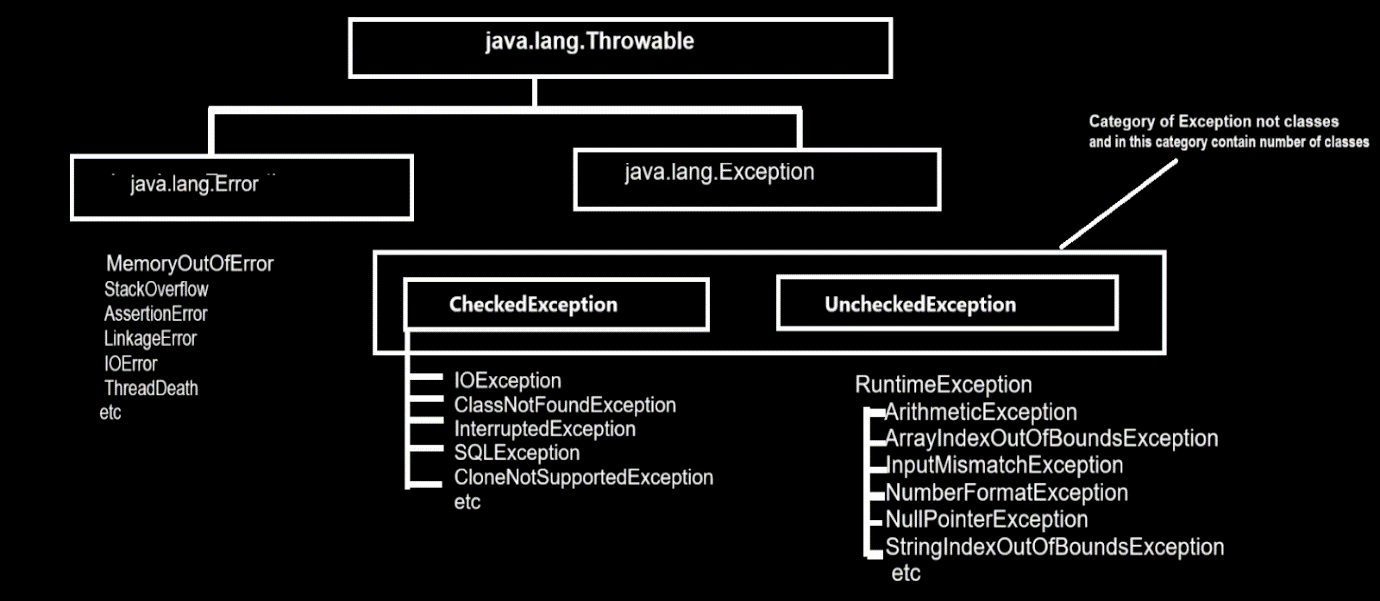
1. **Throw : t**hrow is clause in exception handling which is used with function means we can use throw keyword in function definition for throw exception object from function definition to function calling point and throw clause specially use by developer if developer wants to handle the user define exceptions.

1. **Throws**: throws clause we can use in function definition and it helps us to return exception objects implicitly from function definition to function calling and normally throws clause use by developer for handling checked exceptions.

If we want to handle the exceptions in JAVA then java provides a number of classes to us for handle exceptions or work with exception

**Class Hierarchy of Exception Handling**

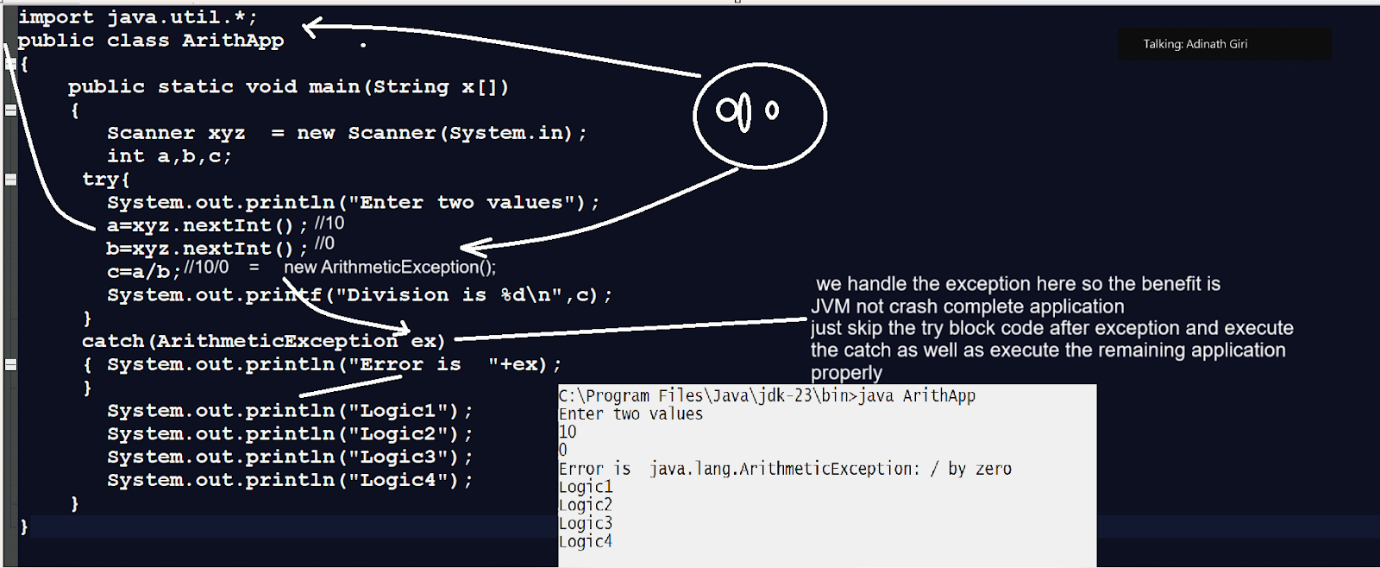
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

**Examples of Exception Handling**

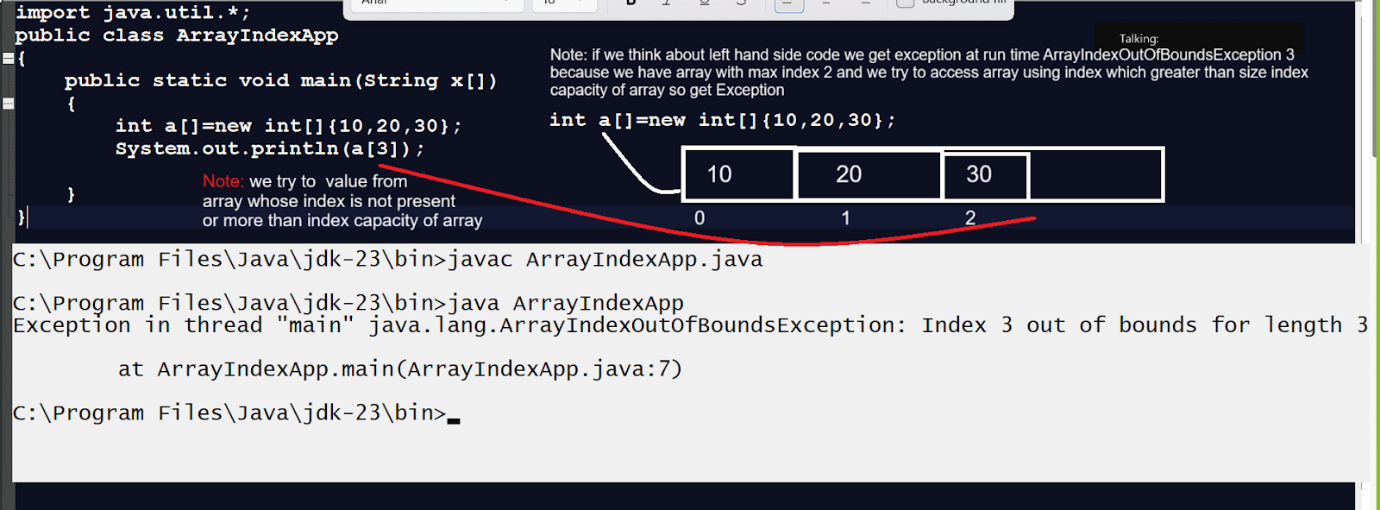
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ArithmeticException:** This is inbuilt class from java.lang package and which is used for handle the division by zero error means when we divide any number by zero then there is infinity result and we cannot calculate infinity result as well as system cannot calculate infinity result so if we have this type of problem and if we want to handle it then JAVA provide ArithmeticException class to us.

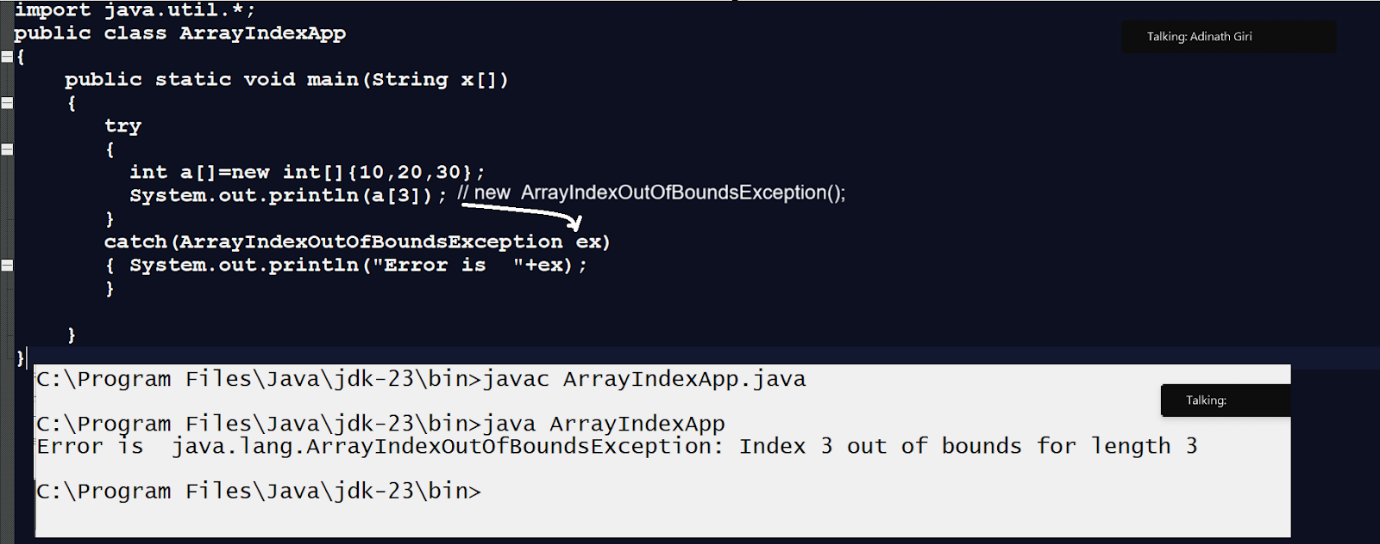
****

**ArrayIndexOutOfBoundsException :** this class is also a member of java.lang package and JVM generates its object as an exception object when developers try to access an array out of its size or range.

So Normally this exception occurs when we try to access at run time or write some logic on an array using its index then there is possible index size may exceed the size of array then JVM can generate the ArrayIndexOutOfBoundsException at run time.

****

**Note:** if we want to handle this exception we have to use the ArrayIndexOutOfBoundsException class from java.lang package shown in below code.

****

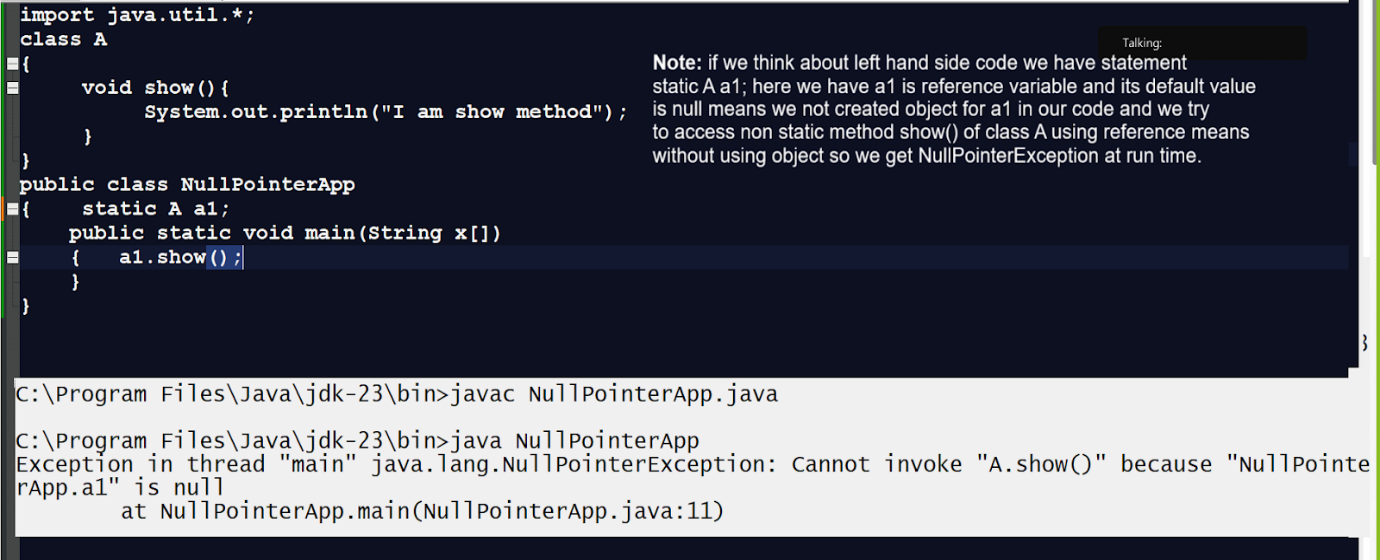
**NullPointerException:** NullPointerException occur when we use any reference without new keyword or without allocation of memory means if we array without new keyword or if we use any object without new keyword then JVM generate the NullPointerException at run time.

****

**Note:** if we want to handle the above exception we have the handle the NullPointerException at program run time using try and catch block.

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

**So use NullPointerException to handle the above code exceptions.**

import java.util.\*;

class A

{

void show(){

  System.out.println("I am show method");

}

}

public class NullPointerApp

{    static A a1;

    public static void main(String x[])

{

  try{

      a1.show();

  }

  catch(NullPointerException ex)

  { System.out.println("Error is  "+ex);

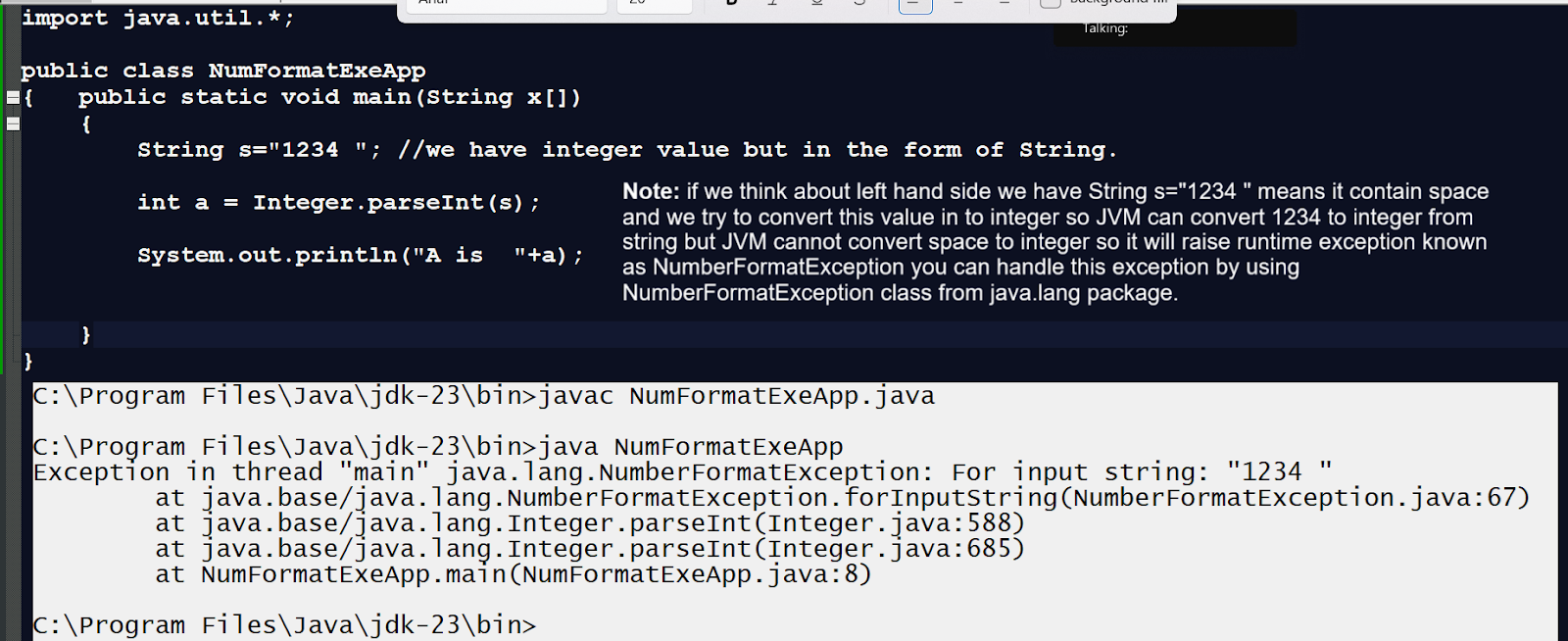
  }

}

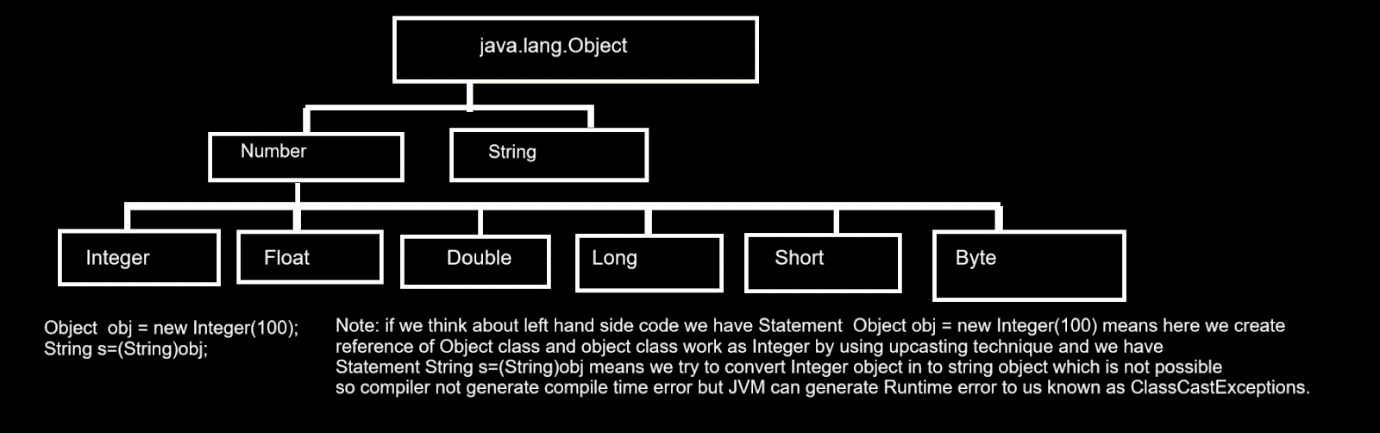
}

**NumberFormatException:** NumberFormatExceptions normally occur when we try to convert string value to primitive type of value.

Sometimes we have numeric values but present in the form of String and we want to convert that value in associated numeric type but if string contains some non numeric values like space or any other alphabet or special symbol then JVM generates the NumberFormatException to us.



**ClassCastException:** Normally ClassCastException occur in Referencial conversion means we have two different types of classes and if we try to convert one class reference to another class reference then there is possible of ClassCastException in JAVA



**FileNotFoundException:** this exception occurs when a file is not  found an  I/O Operation.

import java.io.\*;

public class ReadFileApp

{

    public static void main(String x[])

{

try{

    FileReader fr= new FileReader("D:\\july 2024\\myfilehandling\\buffaprp.txt");

int data;

while((data=fr.read())!=-1)

{

    char ch=(char)data;

System.out.print(ch);

}

}

  catch(FileNotFoundException ex)

  { System.out.println("Error is "+ex);

  }

  catch(IOException ex)

  {System.out.println("Error is  "+ex);

  }

}

}

**IllegalArgumentException: Normally this exception thrown by java developer or JAVA inbuilt API when we try to provide wrong argument to the function/array etc**

**Example:** Suppose we are creating an array and  array size should not be negative and if someone tries to set array size as negative we want to throw IllegalArgumentException at program run time.

import java.util.\*;

public class TestArrApp

{

    public static void main(String x[])

{

try{

  Scanner xyz  = new Scanner(System.in);

  int size;

  System.out.println("Enter size of array");

  size=xyz.nextInt();

  if(size<0)

  {

throw new IllegalArgumentException("Invalid argument pass to array should not negative");

  }

  int a[]=new int[size];

  System.out.println("Length of array is "+a.length);

}

catch(IllegalArgumentException ex)

{ System.out.println("Error is  "+ex.getMessage());

}

}

}

**IndexOutOfBoundsException:** this exception occur access invalid index as list or array or string.

import java.util.\*;

public class TestArrApp

{

    public static void main(String x[])

{

try{

  ArrayList al = new ArrayList();

al.add(10);

al.add(20);

  Object obj = al.get(2);

  System.out.println(obj);

}

catch(IndexOutOfBoundsException ex)

{ System.out.println("Error is  "+ex);

}

}

}

**NoSuchElementException:** this exception occur if we have no element present in collection and try to fetch it then we get NoSuchElementException found

import java.util.\*;

public class TestArrApp

{

    public static void main(String x[])

{

try{

  List al = new ArrayList();

  Iterator i=al.iterator();

  System.out.println(i.next());

}

catch(NoSuchElementException ex)

{ System.out.println(ex);

}

}

}

**SQLException:** this exception normally handles the database connection related exceptions.

**StackOverflow:** this exception thrown by JVM when your stack memory exceeds.

import java.util.\*;

public class TestArrApp

{

    public static void main(String x[])

{

  show();

}

public static void show()

{

try{

  show(); //recursion call

}

catch(StackOverflowError e)

{ System.out.println(e);

}

}

}

**OutOfMemoryError:** this exception occurs when JVM runs out of memory during runtime

import java.util.\*;

public class TestArrApp

{

    public static void main(String x[])

{

try{

    int a[]=new int[Integer.MAX\_VALUE];

}

catch(OutOfMemoryError ex)

{

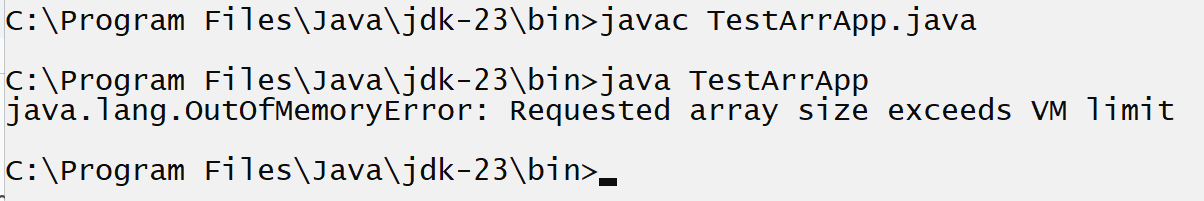
  System.out.println(ex);

}

}

}

**Output**



**NoClassDefFoundError  & ClassNotFoundException:** this error occurs when we use the particular in code but class exists then we get this type of error.

Basically ClassNotFoundException occur at program compile time.

**Example:** Sometime we are creating object of class by using Class.forName() method if class not present then we get this type of exception

import java.util.\*;

public class TestArrApp

{

    public static void main(String x[])

{

try{

Class.forName("asdfaf");

}

catch(ClassNotFoundException ex)

{ System.out.println("Error is "+ex);

}

catch(NoClassDefFoundError ex)

{ System.out.println("Error is  "+ex);

}

}

}

**Finally block**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Finally is block in exception handling which always execute if exception generate in program or not

import java.util.\*;

public class DivApp

{

    public static void main(String x[])

{

Scanner xyz  = new Scanner(System.in);

try{

int a,b,c;

System.out.println("Enter two values");

a=xyz.nextInt();

b=xyz.nextInt();

c=a/b;

System.out.printf("Division is %d\n",c);

}

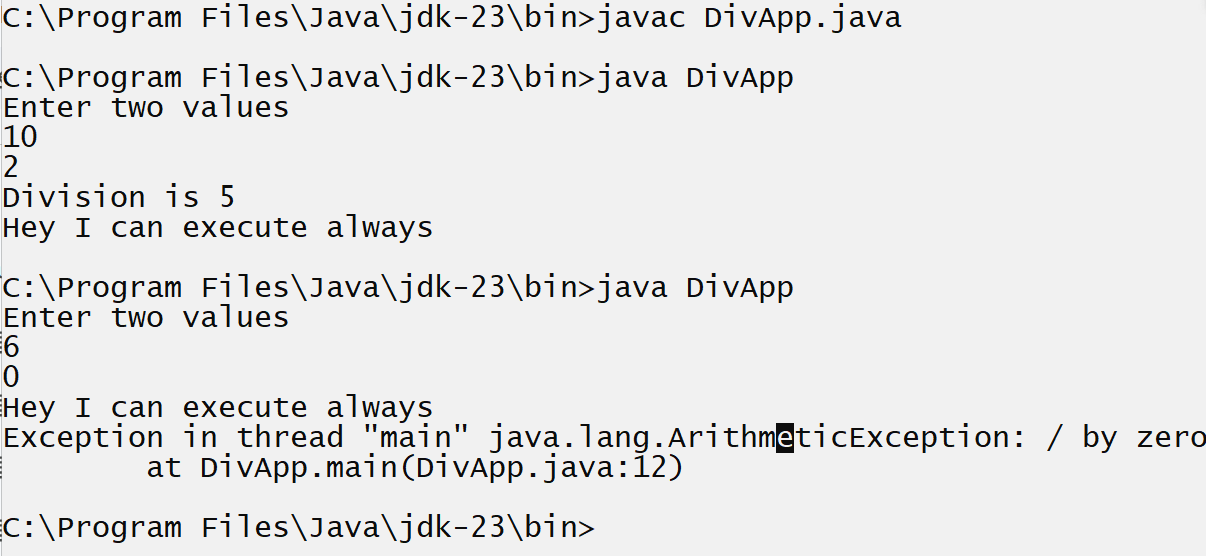
finally

{ System.out.println("Hey I can execute always");

}

}

}



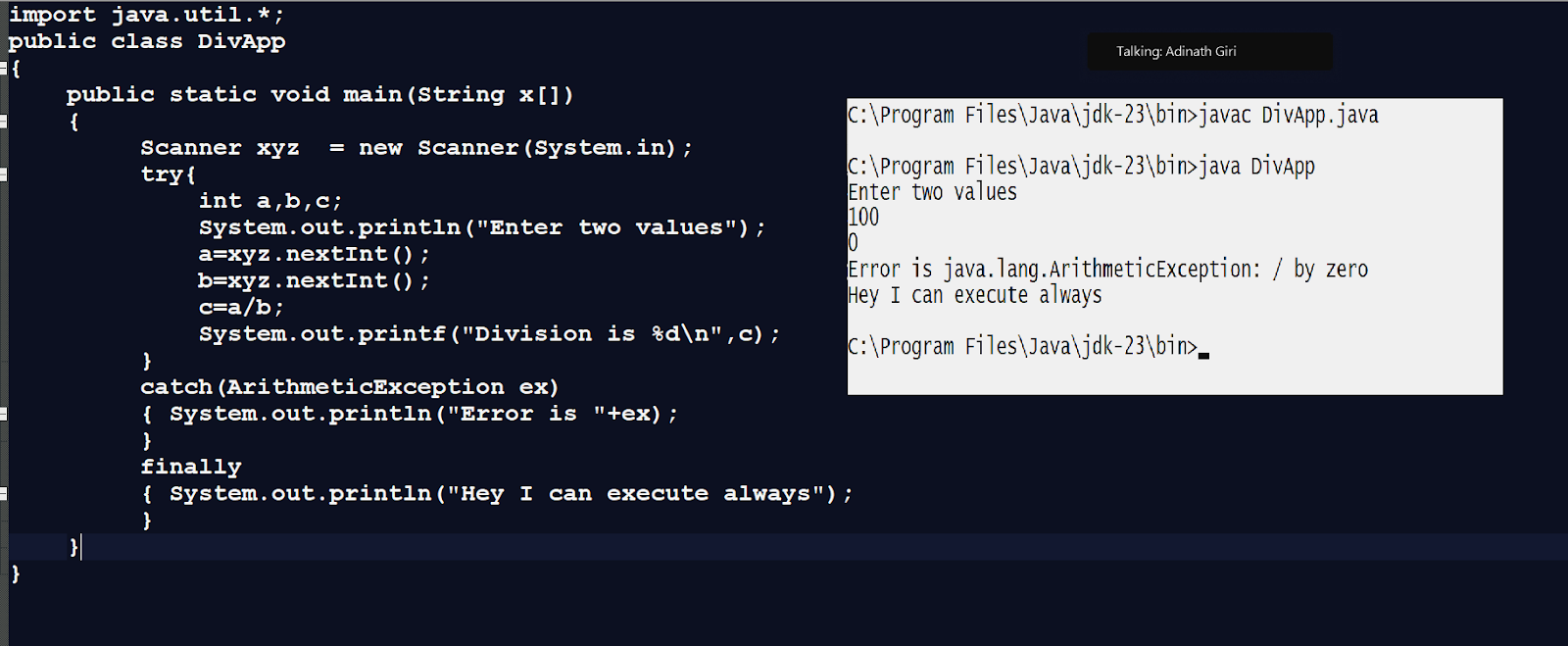
**Note**: if we think about above screen shot we can check finally executed in both situation when our program executed properly means there is exception then finally get executed and if our code has exception then finally executed before exceptions

So we can say finally executed if an exception is generated in code or not.

**Q. What is the difference between catch and finally?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Finally is not responsible for  handle the exception just finally take care own logics means execute own logic if exception occur or not and catch is responsible for handle the exception
2. If use finally and if program contain exception then finally executed before exception and catch execute after exception if occur
3. When we use try ,catch and finally all at the same time then catch has higher execution priority than finally means if an exception is generated in code then catch executes before finally.



**Throw and throws**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Throws :** throws is a clause in exception handling which help us to handle the checked  exceptions

Normally we use throws clause with function definition and when we use throws clause with function definition we need to write try and catch at function calling point.

Normally throws use when we want to handle the exception at function calling point in compilation phase

**Syntax:**

return type functionname(datatype variablename)throws exceptiontype

{

**}**

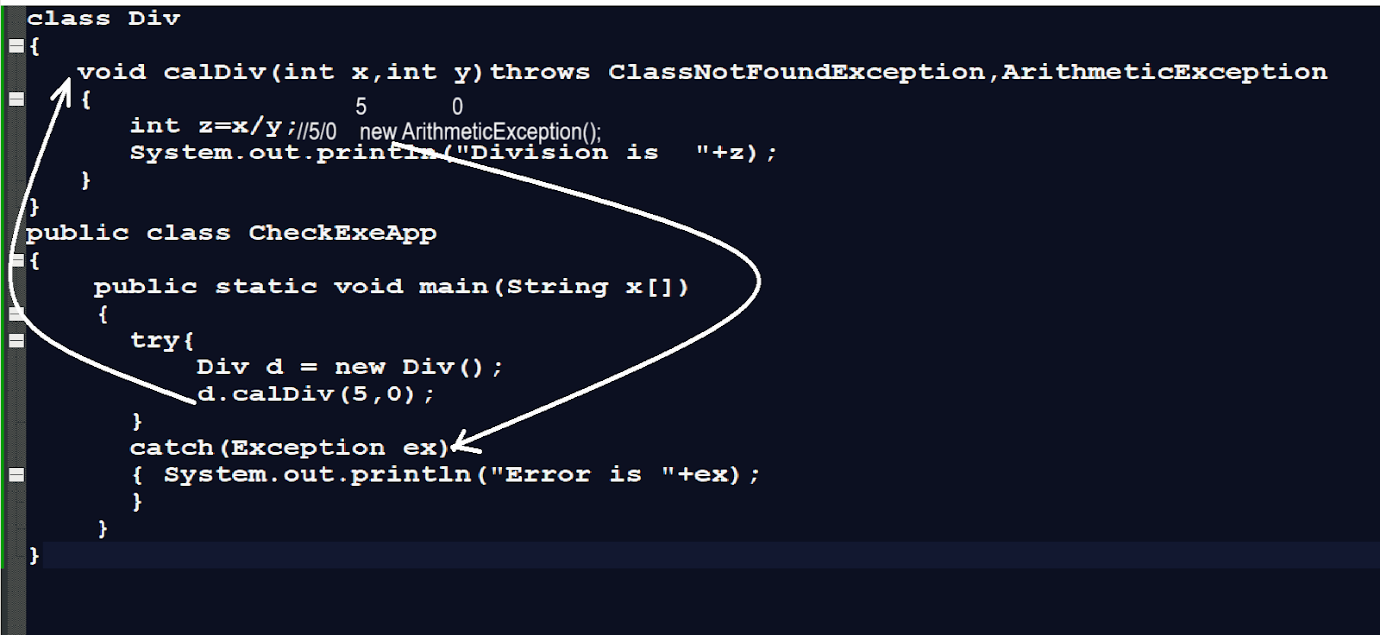
**Note: if we think about throws clause we have some important points.**

1. When we use throws clause we not need to write try and catch in function definition

1. When we use throws then we must have to use try and catch at function calling point or means we must have to handle exceptions at function calling point.

1. When we use throws with function and we do not use try and catch at function calling point then we get compile time error or compiler time exception error.

1. When using the throws then JVM creates an exception class object in function definition and throws it at the function calling point.



**Throw clause**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Throw clause is used to handle the user defined exceptions.

**Q. What are user defined exceptions?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Those exceptions defined by the user for its own use are called as user-defined exceptions.

**Q. Why do users need to develop their own exceptions?**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. If a developer has some exception in an application but java does not provide an inbuilt exception class to handle that exception then the developer can create own exception class called as user defined exceptions.
2. If a developer wants to create exceptions or error messages according to project requirements then the developer can create own exception class called a user defined exceptions.

**How to create user defined exceptions in JAVA?**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

If we want to create user define exception in java we have to create own class and inherit the inbuilt class properties in it

Means using inheritance we can access Exception handling properties define by JAVA as well as we are adding own properties in it according to our requirement

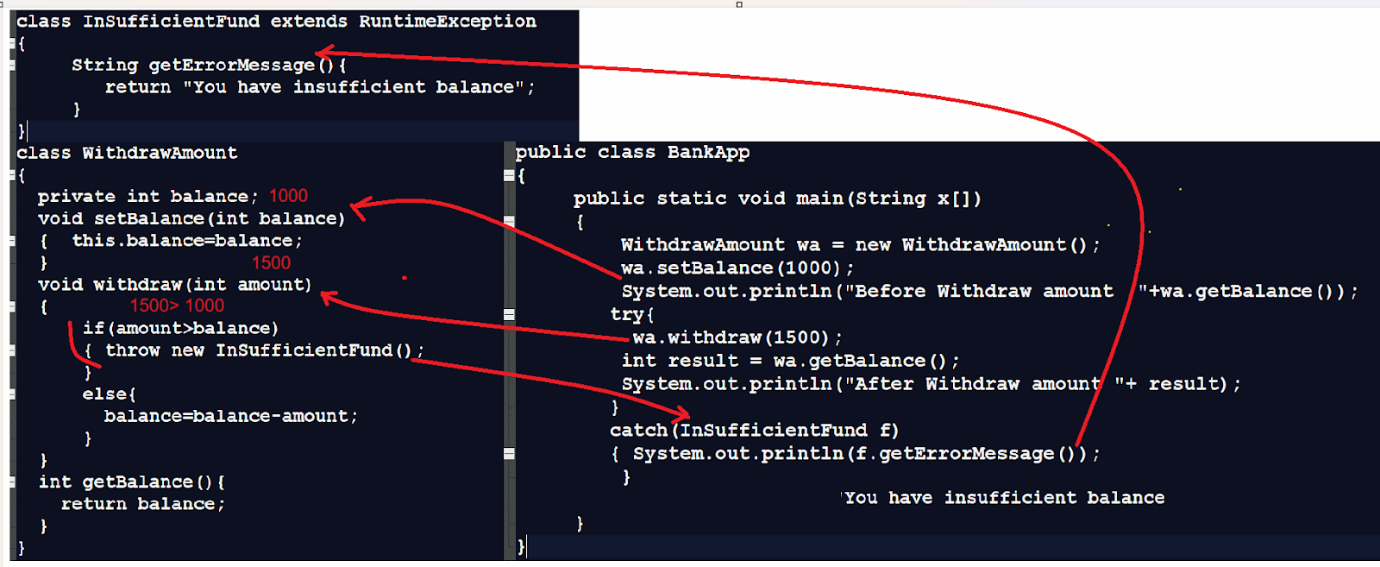
**Syntax:**

access specifier class class name extends exceptionclassname

{

}

**Example**: Suppose consider we are working on Banking Application and user try to withdraw amount more than balance of his account then system should generate the exception at run time insufficient fund then we can use the user define exceptions



**Example:** Suppose consider we are developing project for ManPower Hiring Agency

Means Goal of our project maintain smooth recruitment process but the clause or rule is employee age should be greater than 15

class AgeVerificationExe extends RuntimeException

{  private String message;

   AgeVerificationExe(String message)

   {

      this.message=message;

   }

   public String getErrMessage(){

return message;

   }

}

class Hiring

{

   void checkAge(int age)

   {

     if(age<=15)

{ throw new AgeVerificationExe("Your age is not suitable "+age);

}

else

{ System.out.println("On boarding process start");

}

   }

}

public class ManPowerHiringApp

{

    public static void main(String x[])

{

  try{

Hiring h  = new Hiring();

h.checkAge(10);

  }

  catch(AgeVerificationExe e)

  { System.out.println("Error is "+e.getErrMessage());

  }

}

}

**Important points related with throws clause**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Throw keyword use to handle user defined exceptions
2. Throw keyword can work with single exception class at time
3. Need to create manual object exception class
4. No need to try and catch in function definition, need to write try and catch at function calling point.

**Q. What is the difference between throw and throws?**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| **Throw** | **Throws** |
| Throw use to handle user define exception | Throws use to handle the checked exceptions |
| Throw can work with single exception object at time | Throws can work with more than one exception at time |
| Throw need to create manually exception object and throw it | Throws create implicitly exception object and throw it |
| No need to handle exception at compile time if use throw keyword | Need to handle exception at compile time if use throws keyword |

Assignment

**Interview Question**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q1. What is Exception and why use it?

Q2. Explain the type exception?

Q3. What is checked, unchecked and error exception?

Q4. Explain exception hierarchy?

Q5. what is used for try ,catch,finally,throws and throw?

Q6. explain any 8 exception class examples with description?

Q7. What is the difference between catch and finally?

Q8. What is the difference between final ,finally and finalize()?

Q9. What is throw and throws explained with an example?

Q10. What is the difference between throw and throws?

**Programming Assignment on user defined exceptions?**

**Q1. Check InvalidEmailFormatException**

Description: you have created an application for checking if an email entered by a user is proper or not means email must contain @ and . (dot operator) and after dot operator there must 2 or 3 characters and dot may be repeat  and @ must be only once in email and if this criteria is not  match then system should generate exception at run time InvalidEmailFormatException

**Q2. ProductOufOfStockException**

**Description:** Suppose  consider we are working on InventoryControl Application and we have one module name as StockManagment and we want to store product count in stack if product is less than 0 then system should generate run time exception to us name as ProductOutOfStockException

Q3. InvalidPasswordException

**Description:** Suppose consider we are working on Login Application and we have one class name as PasswordChecker and this class check the password enter by user and password checking rules given below

1. Password length must be minimum 8 char
2. Password must be contain at  least on capital letter
3. Password must be contain at least on digit
4. Password must be contained at one special symbol

**Note:** if any one of above criteria not match in user password then system should generate exception InvalidPasswordException