what about the exception object creation if the constructor gives an exception, we know that the
stacktrace methods are in Throwable class,.....
so how will we get the exception object from constructor?
   Constructor => if exception is generated it will given to main() by JVM

what happens if we are keeping an empty catch block
      It would result in smoothful termination of the code.
    try{
            int a=10/0;
            System.out.println(a);
   }catch(Exception e){
      //handling code
   }

so sir if there is an error and whether or not the catch will match exception or not all the code
outside the finally block will not get executed?
      Ans. yes finally block will be executed even in this case,but not the code outside finally
            block and program would result in "Abnormal Termination".
            if exception occurs and if it is handled, then finally followed by other statments also
            would be executed and program would result in "Smoothful termination".
try{
      //risky code
      Excpetion genereated is ArrayIndexOutOfBoundsException
}catch(ArithmeticException e){
      //handling code
}finally{
      //resource releasing code
}
      //stmt-1
      //stmt-2

sir u said in slides that multiple try blocks cannot present but in the last code multiple try sare there i did not understood the last code snipet sir
      try with catch and finally -> valid
      try and finally -> valid
      try and catch -> valid
      try{

       }catch(XXXX e){

      }finally{

      }

valid
------
try{
      try{

      }catch(XXXX e){

      }
}finally{

```
}
```

if the inner try catch block has thrown exception, inner catch does not matched, and it
matches outer catch block, before going control to outer catch, it will execute both finally block
and followed statement which are present in the inner try catch block?

```
try{
      try{
            //executed and exception occured not handled
      }catch(XXXX e){

      }
      finally{
            //gets executed normally
      }
      //will not be executed
      stmt-1
      stmt-2;
}catch(YYYY e){
      //exception handled
}finally{
      //statment executed
}
```

can you tell us again throw keyword? How the flow go the throw e?
        throws -> it is normally used with checkedException.
                  CheckException -> compiler will scan the code and it will check wheter the
                                                    Exception would occur or not at the runtime.

            eg: IOException,SQLExeption,....

      throw -> It is normally associated with UnCheckedException
                  CheckException -> compiler will scan the code and it will not check wheter the
                                                Exception would occur or not at the runtime.
                                                It is the duty of the programmer to make sure the termination
                                                should happen normally.

sir how it is possible to use return keyword in try block? as per my knowledge return is used to
transfer control back to caller method. plz explain.

```
public class Test{

      public int m1(){
          try{
            return 10;
          }catch(Exception e){
                  //handling logic
         }finally{
                  stmt-1;
         }
```

```
      }
}
```

sir if checked exception are not handled then it will be handled by DEH in runtime yes or no sir

Sir please can you explain throw keyword not cleared

```
   BankManager
         |
      Account
      public String deposit(){
                  //exception occured in the balance part for this customer
                  //exception should be handled
                  throw new problemInDepostingMoneyException();
      }
      public String changeName(){
                  //exception of changing Name
                  //exception occured and it is handled by Account class only.
        }
          |
       Customer
```

sir if checked exception are not handled then it will be handled by DEH in runtime yes or no sir

JDBCApi code(not written by end user)

```
==============================
public class DriverManger{
      public static Connection getConnection(String url,String username,String
password) throws SQLException
}

public class DemoApp{
      public static void main(String[] args) throws SQLException{
            Connection connection =
DriverManger.getConnection("jdbc:mysql://localhost3306/demo","root","root123");
      }
}
```

Exception had not occured => then connection will happen for MySQL database
Exception occured => JDBC Api will throw SQLException to JVM, JVM will check whether main()
has the handling code or not.
if handling code exists inside main then smoothful termination of application
if handling code does nt exists then JVM will delegate that SQLException object
to its own "Handler" called "DefaultExceptionHandler" and program would result
in "Abnormal termination".

sir we know that finally has cleanup code is it possible that finally can also generate exception, then in that case resource is stuck right?
then what to do?

```
      //Resource used in try block
```

```
      Connection con = null;
      try{
            //risky code
            con=DriverManager.getConnection(url,username,password);
      }catch(Exception e){
            //handling logic
      }finally{
            //resource releasing logic
            if(con!=null)
                  con.close();
      }

int method1(){
try{
      sop("");
      return 10;
}
finally{
      SOP(inside finally)
}

}
```

what will be the flow and whether again and again flow will enter into try just for return...

what is logical difference in ducking a exception and rethrowing an exception as they are giving same result

ducking -> u r just using throws keyword(no handling logic)

reThrowing-> we are handling it and also informing the caller that Exception occured(handling logic is available).

sir my another question will be as you told compiler will automatilaay initialize inbuild class object as
arithmatic excetion - new arithmaticex(); how compiler do that ? and what in case of custom exceptions?

Compiler => it will just whether the code is generating any checkedException or not and if yes it will also
check whether the programmer has wrote the handling logic or not.
hanlding logic can be either try{} catch(){} or using throws keyword

```
      try{
            int a=10/0; //JVM will execute this and it throws and Exception called "ArithmeticException".
                              //new ArithmeticException("/ by Zero");
      }catch(Exception e){//Exception e =new ArithmeticException("/ by Zero");

      }

public class A {

    public static void main(String[] args) {

        try{
            System.out.println("In try Block A");/stmt will be executed
            int c=10/0;//new ArithemeticExceptin("/ by zero");
```

```
        }catch (NullPointerException e){
            System.out.println("Null Pointer Exception");
        }finally {
            System.out.println("Finally Block A");//stmt will be executed
            try{
                System.out.println("In try Block B");//stmt will be executed
                int d=10/0;//new ArithemeticExceptin("/ by zero");
            }catch (ArithmeticException e){
                System.out.println("Arithemetic Exception");//stmt will be executed
            }
            System.out.println("outside catch block finally");//stmt will be
executed
        }

        System.out.println("I am outside of try and catch");
}
```

sir exception occured at inner catch ,please explain sir statements would be executed??

```
statement 1;
try
{
      statement 2;
      statement 3;
      statement 4;
      try{
            statement 5;
      }catch(xx e){
            statement 6;
      }
      statement 7;
}
catch(xy e)
{
      statement 8;
}
finally
{
      statement 9;
}
```

1,2,3,4,5,8,9(smoothful termination) if exception is handled.
1,2,3,4,9(abnormal termination) as excepition is not handled.

```
package Exception;

import java.util.InputMismatchException;
import java.util.Scanner;

public class InnerExc
{
    public static void main(String[] args) {
        try {
            Scanner scan = new Scanner(System.in);
            System.out.println("enter first number for division");
            int a = scan.nextInt();
            System.out.println("enter second  number for division");
            int b = scan.nextInt();
            try {
```

```
                        int c = a / b;
                        System.out.println(c);
            }catch (InputMismatchException ex){
                        ex.printStackTrace();
            }
            finally {
                        System.out.println("inner finally block");
            }
             System.out.println("inner loop");
           }catch (ArithmeticException ex){
               ex.printStackTrace();
           }finally {
               System.out.println("out side finally block");
           }
           System.out.println(" in main method");
    }
}//when i execute this code if i enter wrong input  InputMismatchException has to
caught but its

//Compile Time error
try{
      finally{
            statement 2;
                  try{
                        statement; //error occured
                  }catch(xx e) //not matched
                  {
                        statement1;
                  }

      }
} catch(exception e)//exception matched {
      sop("Hi")
 }

what is the extra use of new ArithmaticException("/ by zero") when jvm already
preparing object, for only custom msg?
    Exception=> They are Objects in java.
                  If problem occurs in our code immediately jvm will generate the
suitable object
                  for the problem.


While ducking compiler will check for exception but rethrow user has to do it what
does it meant by sir?
      ducking -> compiler warns u to write the handling code for the  statements

JDBCApi code(not written by Oracle developers)
==============================
public class DriverManger{
      public static Connection getConnection(String url,String username,String
password) throws SQLException
}

Progammer code
============
public class DemoApp{
      public static void main(String[] args) throws SQLException{
            Connection connection =
```

```
DriverManger.getConnection("jdbc:mysql://localhost3306/demo","root","root123");
      }
}


Code
===
try{
            System.out.println("In try Block A");//stmt will be executed
            int c=10/0;//new ArithmeticException("/ by Zero")
    }catch (NullPointerException e){
            System.out.println("Null Pointer Exception");
    }finally {
            System.out.println("Finally Block A");//stmt will be executed
            try{
                System.out.println("In try Block B");//stmt will be executed
            }catch (ArithmeticException e){
                System.out.println("Arithemetic Exception");
            }
            System.out.println("outside catch block finally");//stmt will be
executed
  }
      System.out.println("I am outside of try and catch");
 }
}

NullPointer Exception
==============
String name=null;
System.out.println(name.length());//NullPointerException

int arr[]=null;
arr[4]=10; //NullPointerException
will get null pointer exception

sir can you give overview of jar vs war file just basic overview?
    JAR -> collection of .class files(jdk s/w which we install is actually .class
files only)
                  It is used when we work with standalone applications.(JSE)

   WAR-> collection of .class files + html files +css files  +javascript files
                  It is used when we work with webapplications/distrubuted
applciation(JEE)


try{
            int res=10/0; // throw new ArithemeticException("/ by zero");
}catch(NullPointerException e){

}

custom exception
============

class StudentRecordNotFoundException extends RunTimeException{
            public StudentRecordNotFoundException(String msg){
                        this.msg=msg;
            }
}
```

```
class TestApp{
      public Student getRecord(String id){
            Student std = get the record from the database
            if(std!=null)
                  throw new StudentRecordNotFoundException("record not found");
            else
                  return std;
  }
}

class Demo{
      public static void main(String...  args){
            try{
                  Student student = new TestApp().getRecord(10);
            }catch(StudentRecordNotFoundException se){
                        se.printStackTrace();
            }catch(Exception e){
                  e.printStackTrace();
            }
       }
}




try{
      stmt-1
      stmt-2
            try{
                  stmt-3//exception occured
            }catch(XXXX e){//exception not handled
                  stmt-4
            }finally{
                  stmt-5
            }
}catch(YYYY e){//exception not handled
            stmt-6
}finally{
            stmt-7
}
```

finally block will be executed and JVM will handled by using DEH.