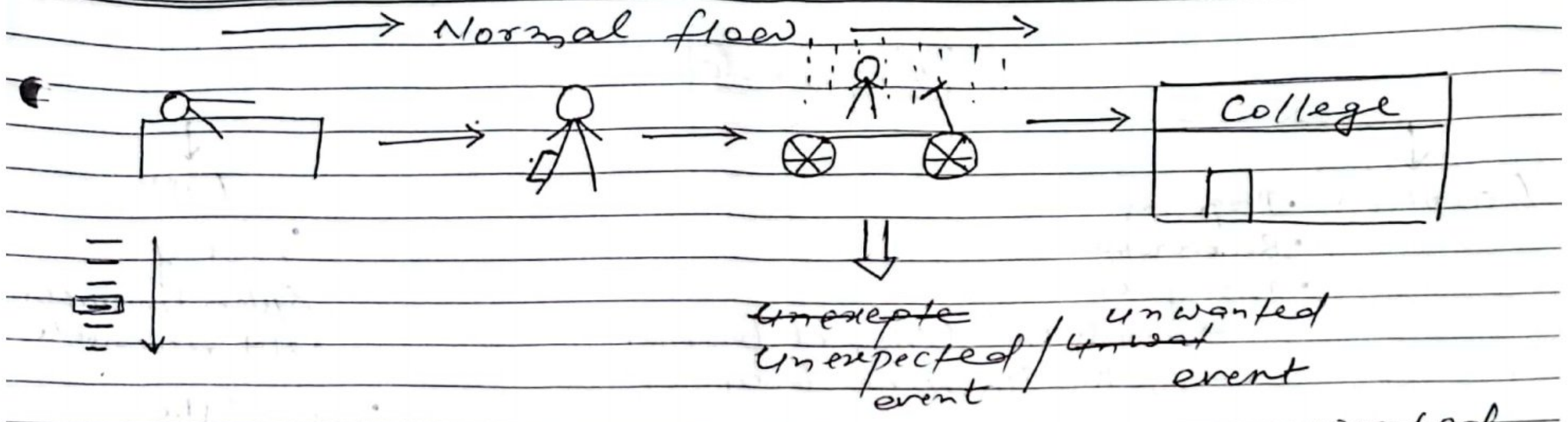
Exception Handling



* An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e. at run time, that disrupts the normal flow of the program.

Object

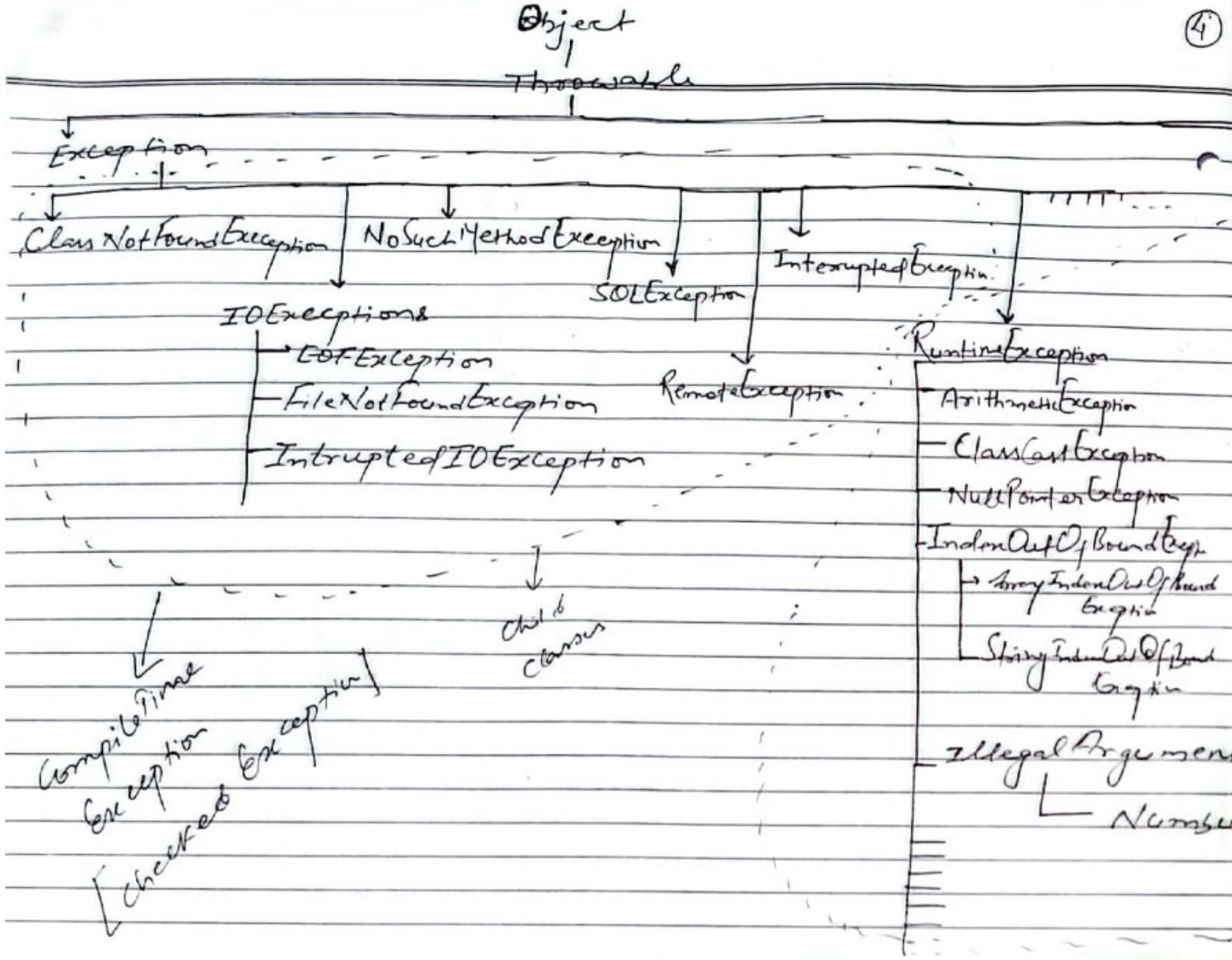
Throwable

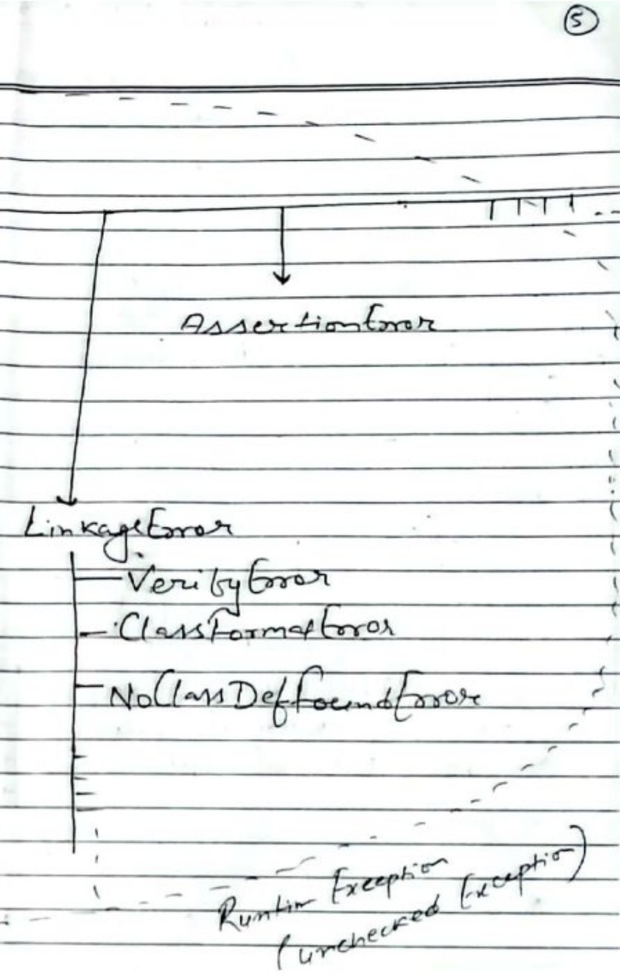
Exception
- Programs
- Recoverable
- Two types
  - → Compile time (Checked Exception)
  - → Run time (Unchecked Exception)

Error
- lack of System Resources
- Not recoverable
- one type
  - → Run Time Exception (Unchecked Exception)

Difference between Exception & Error

Exception	Error
1. Exception occurs because of our programs.	1. Errors occurs because of lack of system resources.
2. Exceptions are recoverable. i.e. programmer can handle them using try-catch block.	2. Errors are not recoverable. i.e. programmer can handle them to their level.
3. Exception are of two types: ↳ Compile time exceptions or checked exceptions. ↳ Runtime exceptions or unchecked exceptions.	3. Errors are only of one type: → Run time Exceptions or Unchecked Exceptions.

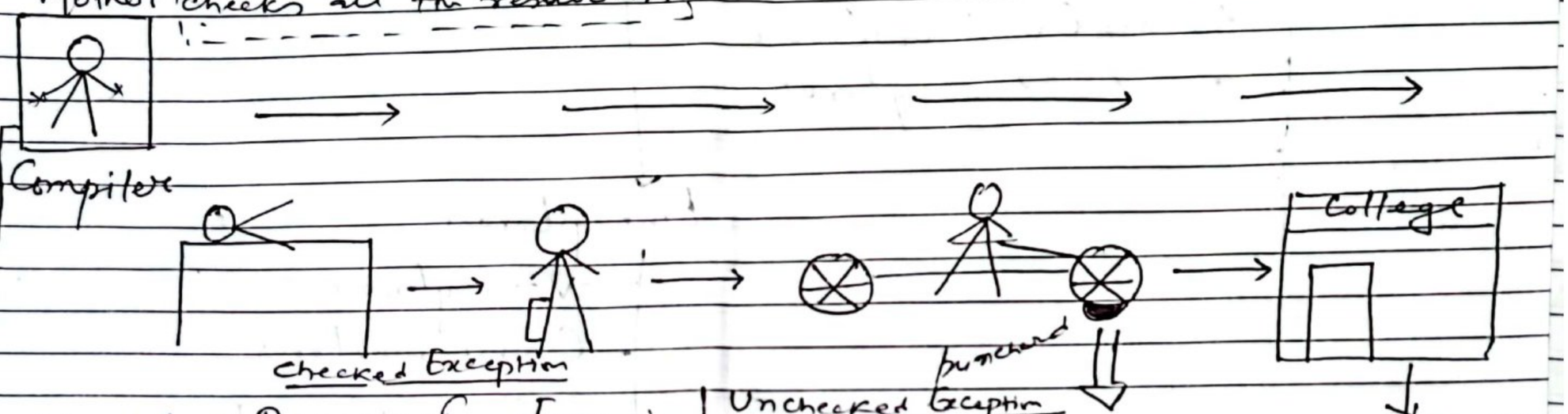

Object
Throwable
Exception
ClassNotFoundException
NoSuchMethodException
SQLException
InterruptedException
IOExceptions
EOFException
FileNotFoundException
InterruptedIOException
RemoteException
RuntimeException
ArithmeticException
ClassCastException
NullPointerException
IndexOutOfBoundsExp.
ArrayIndexOutOfBound Exception
StringIndexOutOfBound Exception
IllegalArgumentException
NumberFormatException
Child Classes
Compile time exception
Checked exception


Error
VirtualMachineError
StackOverFlowError
OutOfMemoryError
AssertionError
LinkageError
VerifyError
ClassFormatError
NoClassDefFoundError
Runtime Exception
Unchecked Exception

↳ 2 ↳ Difference between Checked ad Unchecked Exceptions:

* Compile time exceptions occurs at Compile time and rem time exceptions occur at rem time.
↳ 'It is a wrong Concept'.

→ Right Concept:
↳ All the exceptions occurs at runtime.



↳ Cheers: → PursenotFoundException
→ CompanyIdNotfoundException
→ LunchBoxNotFoundException

Unchecked Exception
→ BikePuncherException Unwanted
→ AccidentOccurException event
→ TrafficFoundException

* All the exceptions are occured at run time only.

* It never occurs at a Compile time.

[Program example] →

// Program Example

```
import java.io.FileInputStream;
class Test
{
public static void main(String[] args)
{
FileInputStream fis = new FileInputStream("d:/abc.txt");
}
}
```

→ Compile the program →
→ Unreported Exception [FileNotFoundException]
(Because Exception is not handled)

→ It is just a warning. but not an Exception
→ It is an error

// Reporting the Exception.

```
import java.io.FileInputStream;
class Test
{
public static void main(String[] args)
```

```
}
try
{
FileInputStream fi = new FileInputStream
                        ("D:/abc.txt");
}
catch (Exception e)
{
System.out.println(e);
}
}
```

output

NO error but Exception is shown

PAGE
DATE

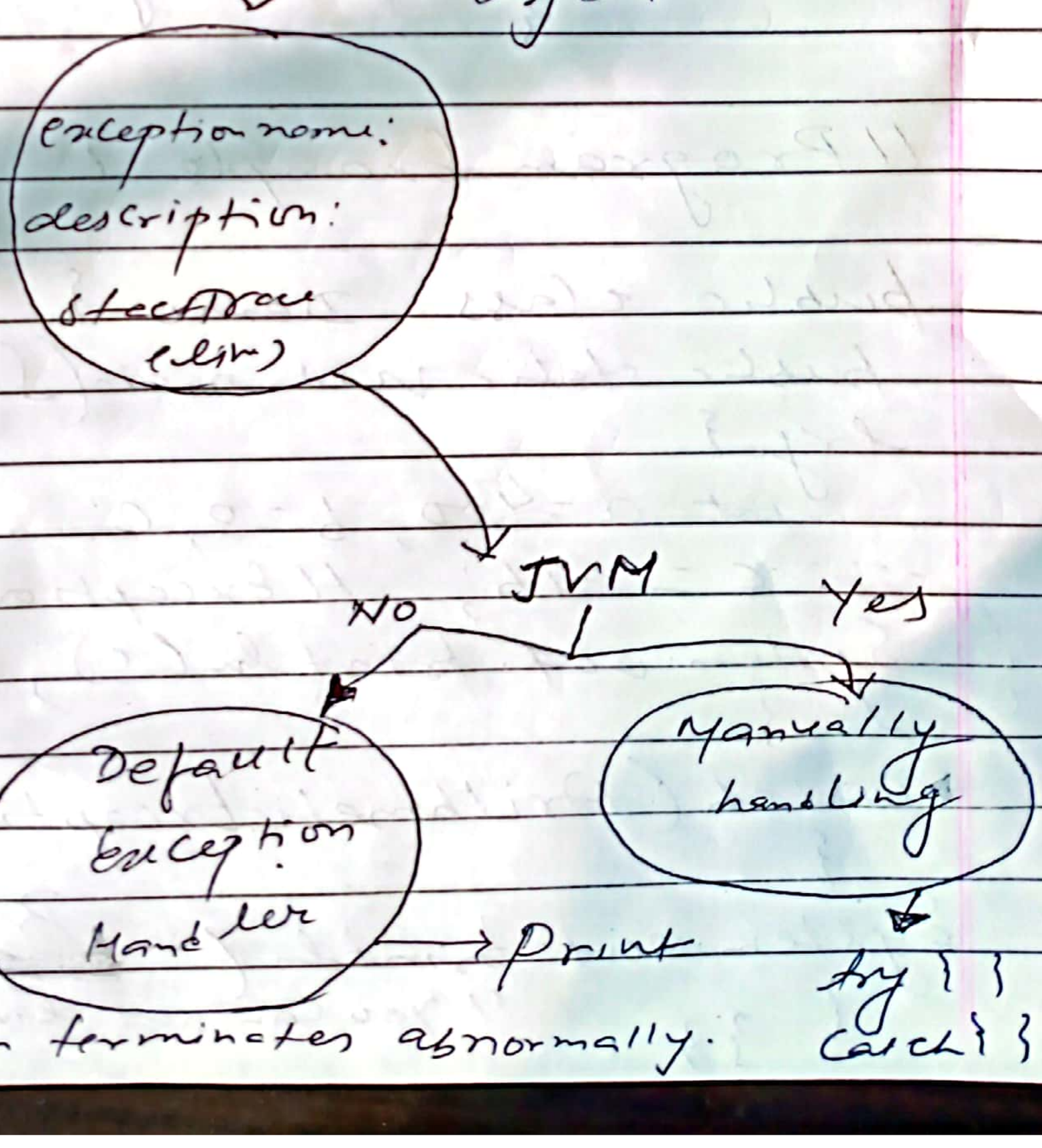
* whenever there is exception, the method in which exception occurs will create an object and that object will store three things :-

1. Exception Name 2. Description
3. stack strace(line)

// Program Example

```
public class Demo {
    public static void main(String[] args) {
        int a=100, b=0, c;
        c = a/b; // Exception occurs
        System.out.println(c);
    }
}
```

// main methods create an object



and program terminates abnormally.

PAGE
DATE

we can handle the exception using 5 keywords:

1. try 2. catch 3. finally 4. throw 5. throws

1. try - catch:

Syntax:

```
try {
    // risky code;
}
catch (ExceptionClassName ref_var)
{
    // handling code;
}
```

// Program Example 1

```
public class Demo {
    public static void main(String[] args) {
        try {
            int a=400, b=0, c;
            c = a/b; // Exception occurs
            System.out.println(c);
        }
        catch (ArithmeticException e) // exception
        {
            System.out.println("Problem: " + e);
            // You can not divide by zero.
        }
    }
}
```

PAGE
DATE

// Program Example 2

```
public class Test {
  public static void main(String[] args) {
    System.out.println("1");
    try {
      System.out.println("2");
      int a=100, b=2, c;
      System.out.println("3");
      c=a/b;
      System.out.println("4");
      System.out.println(c);
      System.out.println("5");
    }
    catch(ArithmeticException e)
    {
      System.out.println("6");
      System.out.println(e);
      System.out.println("7");
    }
    System.out.println("Last statement");
  }
}
```

PAGE
DATE

output:

```
1
2
3
6
java.lang.ArithmeticException
: / by zero
}
Final Statement
```

Note: Statement '4' is not executed because it is 'below' of exception statement.

C:\Users\student\Desktop\Demo.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Demo.java | Demo.java

```
1  //run time exception
2  //exception is not handeled(Unchecked exception(
3
4  public class Demo
5  {
6      public static void main(String[] args)
7      {
8      int a=121,b=0,c;
9      c=a/b;//exception
10     System.out.println(c);
11     }
12 }
13
```

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop>javac Demo.java

C:\Users\student\Desktop>java Demo
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Demo.main(Demo.java:9)

C:\Users\student\Desktop>
```

ava source file | length : 213 lines : 13 | Ln : 13 Col : 1 Pos : 214 | Windows (CR LF) UTF-8 | INS

Type here to search | 74°F Haze | 9:42 PM 12/27/2022

```java
public class Demo
{
    public static void main(String[] args)
    {
    try{
    int a=121,b=0,c;
    c=a/b;//exception
    System.out.println(c);
    }
    catch(Exception e)
    {
        System.out.println("Error:"+e);
    }
    }
}
```

```
C:\Windows\System32\cmd.exe

C:\Users\student\Desktop>javac Demo.java

C:\Users\student\Desktop>java Demo
Error:java.lang.ArithmeticException: / by zero

C:\Users\student\Desktop>
```

ava source file length : 290 lines : 20 Ln : 11 Col : 27 Pos : 212 Windows (CR LF) UTF-8 INS

Type here to search 74°F Haze 9:45 PM 12/27/2022

C:\Users\student\Desktop\Demo.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Demo.java | Demo.java

```
1  //run time exception
2  //exception is not handeled(Unchecked exception(
3
4  public class Demo
5  {
6      public static void main(String[] args)
7      {
8      try{
9      int a=121,b=0,c;
10     c=a/b;//exception
11     System.out.println(c);
12     }
13     catch(Throwable e)
14     {
15         System.out.println("Error:"+e);
16     }
17     }
18 }
19
20
```

C:\Windows\System32\cmd.exe

```
C:\Users\student\Desktop>javac Demo.java

C:\Users\student\Desktop>java Demo
Error:java.lang.ArithmeticException: / by zero

C:\Users\student\Desktop>
```

va source file | length : 290 lines : 20 | Ln : 15 Col : 20 Pos : 256 | Windows (CR LF) UTF-8 INS

Type here to search

74°F Haze

9:46 PM 12/27/2022

CamScanner

Demo.java

```java
//Compile Time Exception(Checked Exception)
import java.io.*;
public class Demo
{
    public static void main(String[] args)
    {
        FileInputStream fin=new FileInputStream("d:\\abc.txt");
        System.out.println("hello World");//will not be executed


    }
}
```

Java source file

C:\Windows\System32\cmd.exe

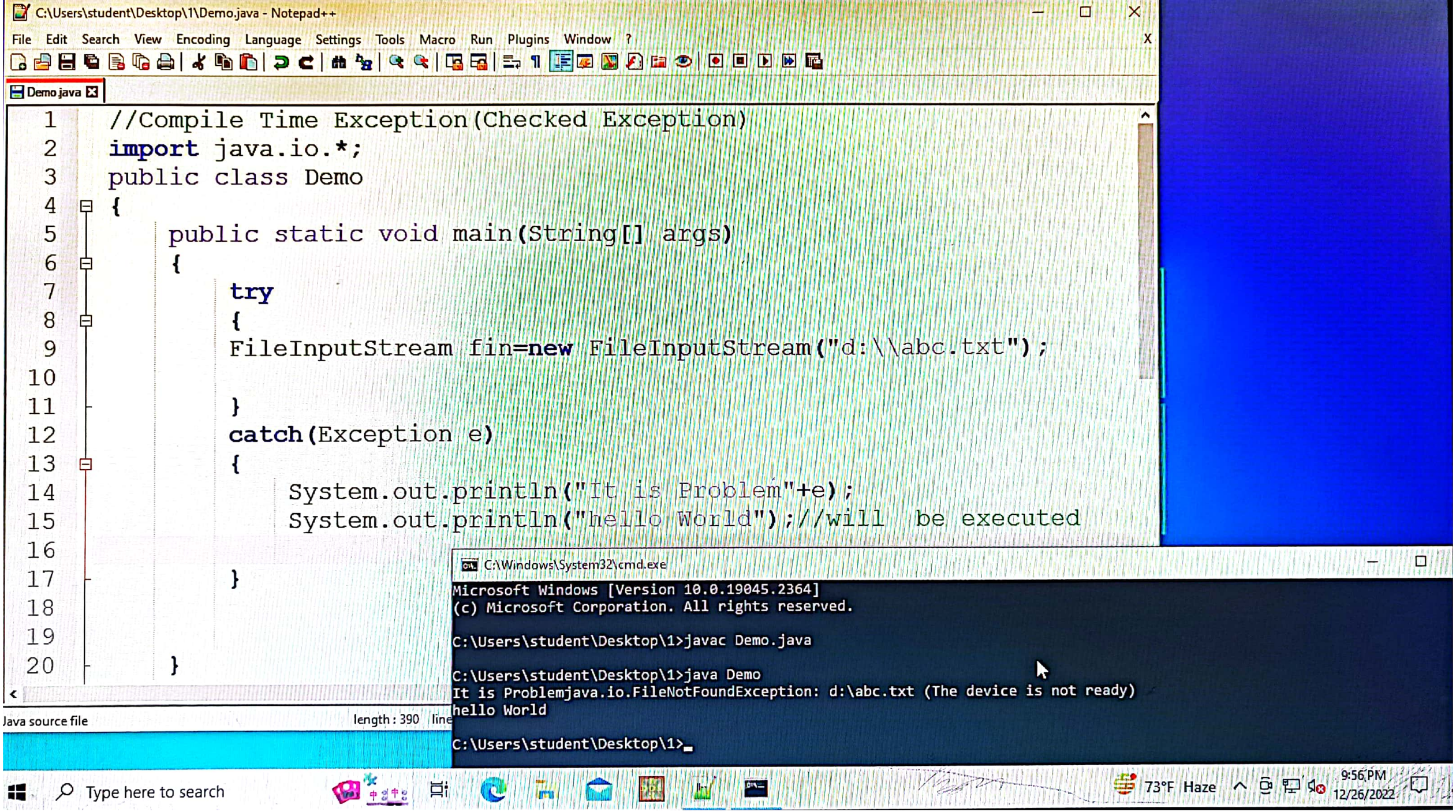
```
Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\1>javac Demo.java
Demo.java:7: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
                FileInputStream fin=new FileInputStream("d:\\abc.txt");
                                    ^
1 error

C:\Users\student\Desktop\1>
```

REDMI NOTE 11 PRO

ASHISH ASHK


C:\Users\student\Desktop\1\Demo.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demo.java
1 //Compile Time Exception(Checked Exception)
2 import java.io.*;
3 public class Demo
4 {
5 public static void main(String[] args)
6 {
7 try
8 {
9 FileInputStream fin=new FileInputStream("d:\\abc.txt");
10
11 }
12 catch(Exception e)
13 {
14 System.out.println("It is Problem"+e);
15 System.out.println("hello World");//will be executed
16
17 }
18
19
20 }
Java source file
length : 390
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.
C:\Users\student\Desktop\1>javac Demo.java
C:\Users\student\Desktop\1>java Demo
It is Problemjava.io.FileNotFoundException: d:\abc.txt (The device is not ready)
hello World
C:\Users\student\Desktop\1>
Type here to search
73°F Haze
9:56 PM
12/26/2022

* Exception class various methods:

```
class Test
  {
public static void main(String[] args)
        {
    try
    {
  int a=50, b=0, c;
   c=a/b;
System.out.println(c);
    }
  catch(ArithmeticException e)
    {
    --
    --
    --
    }
}
}
```

Created by default Exception Handler with all detailed Exception

- Exception name
- description
- stacktrace

// different methods used in Catch block

1. e.printStackTrace();
   ↳ ExceptionName ↳ description ↳ stackTrace

2. System.out.println(e);
   or
   System.out.println(e.toString());
   ↳ ExceptionName ↳ description

3. System.out.println(e.getMessage());
   ↳ description

Name : Class :

// Program example

```
public class Test
    {
public static void main (String[] args)
    {
  try {
      int a = 500, b = 0, c;
      c = a / b;
System.out.println(c);
    }
catch ( ArithmeticException ae)
    {
  // ae.printStackTrace();
  // System.out.println(ae);
  // System.out.println(ae.toString());
  System.out.println (ae.getMessage());
    }
  }
```

** finally block :-

The finally block in java used to put important codes (cleanup code) e.g: closing the file or closing the connection.

The finally block executes whether ~~exceed~~ exception rise or not and whether exception handled or not.

A finally block contains all the crucial statements regardless of the exception occurs or not.

Syntax:

```
try
  {
    //risky code
  }
catch(Exception e)
  {
    //handling code
  }
finally
  {
    //clean-up code
  }
```

xx

```
try
  {
    -
    -
  }
finally
  {
    -
    -
  }
```

✓

```
finally
  {
    -
    -
  }
```

✗

```
catch e --- )
  {
  }
finally
  {
  }
```

✗

Note: finally block executes its self where there is exception or not.

Name : Class :

// Program Example

```
public class Test
    {
public static void main(String [] args)
    {
try
    {
    int a = 5, b=0, c; // b = 5
System c = a/b;
    System.out.println(c);
    }
catch (Exception e)
    {
e.printStackTrace();
    }
finally {
System.out.println("It is finally block");
    }
}
}
```

Note: Here, at b = 0 or b = 5, in both cases finally block gets executed.

→ It is mainly used for file handling concepts

XX Multiple "Catch" block:

Syntax:

```
try {
    catch (ExceptionClassName object)
    {
    }
catch (ExceptionClassName object)
    {
    }
    .
    .
    .
```

// Program Example

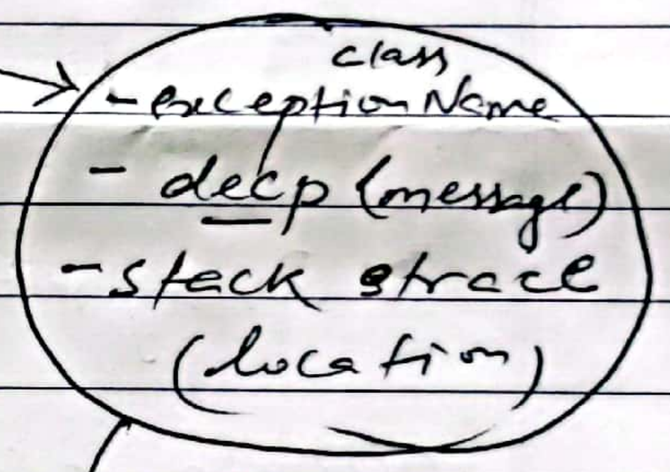
```
public class MultipleCatchBlock1 {
    public static void main(String[] args) {
        try {
            int a[] = new int[5];
            a[5] = 30/0;
        }
        catch (ArithmeticException e) {
        System.out.println("Arithmetic exception occurs");
        }
        catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("ArrayIndexOutOfBounds
                exception");
        catch (Exception e)
                {
```

Name : Class :

```
        System.out.println("Parent Exception occurs");
        }
        System.out.println("Rest of the code");
    }
}
```

** 'throw' Keyword in Java exception :

```
public class Test
    {
public static void main(String[] args)
        {
int a=5, b=0, c;
  c = a/b;
System.out.println(c);
    }
}
```



→

```
class Test
    {
public static void main(String[] args)
        {
Test t = new Test();
t.divide();
}
```

```
public void divide()
        {
int a=5, b=0, c;
c = a/b;
System.out.println(c);
    }
}
```

→ exception
classname
→ description
(message)
→ Stack Trace
(Line no)

throw keyword is used to create a ~~custom~~ custom error. The throw statement is used together with an exception type.

There are many exception types available in java:
ArithmeticException, ClassNotFoundException etc.

↳ It is used for user defined / custom exception.

Syntax:

```
throw new ExceptionClassName("...");
```

// Program Example

```
class YoungerAgeException extends RuntimeException
{
  YoungerAgeException(String msg)
      {
        super(msg);
      }
}
```

Name :

Class :

```
public class voting
    }
public static void main(String [] args)
        {
    int age = 16;
if (age < 18)
    {
throw new YoungerAgeException("you are
        not eligible for vote");
    }
else
    {
    System.out.println("vote success");
    }
}
```

- Ex. classname
- desc
- stack strace

Object

Note :→ throw keyword creates an exception object manually (by programmer) and handover to jvm.

→ we can throw either checked or unchecked exception but throw is best for customized exception.

→ we can only throw class that comes in throwable child class.

→ we can not write any statement after throw, otherwise it will provide

~~can~~ unreachable statement error.

~~**~~ ~~broad~~

```
// Handling Exception
import java.util.Scanner;
class YoungerAgeException extends
                        RuntimeException
{
YoungerAgeException(String msg)
{
super(msg);
}
}

public class vote
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter your age:");
int age = sc.nextInt();
try {
if(age<18)
{
throw new YoungerAgeException(
"you are not eligible for voting");
}
```

Name :

Class :

```
else
    {
System.out.println("you can vote
                    successfully");
    }
}
catch(YoungerAgeException e)
    {
    e.printStackTrace();
    }
System.out.println("hello"), //executed
    }
}
```

** throws Reyword:

throws keyword is used to declare an exception. It gives an information to the called method that there may occur an exception so it is better for the caller method to provide the exception handling code so that normal flow can ~~be~~ to be maintained.

// Program example

```
import java.io.*;

class ReadAndWrite
    {
void readFile() throws FileNotFoundException
    {
FileInputStream fis = new FileInputStream("d:/
                                    abc.txt");
    }
void saveFile() throws FileNotFoundException
    {
FileOutputStream fos=new FileOutputStream("
                    d:/xyz.txt");
    }
}

public class Test {
public static void main(String[] args)
    {
ReadAndWrite rw=new ReadAndWrite();
try {
    rw.readFile();
    }
catch (FileNotFoundException e)
    {
e.printStackTrace();
    }
System.out.println("Hello"); //executed
```

Name :

Class :

~~Note:~~

try {

rw.savefile();

}

Catch (FileNotFoundException e)

{

e.printStactTrace();

}

}

}

Note:

→ throws keyword is used to declare only for the checked exceptions.

~~If there occurs any unchecked exception such as nullpointer~~

Done

Assignment:

↳ difference between throw and ~~throw~~ throws keywords

↳ difference between final and finally keywords.