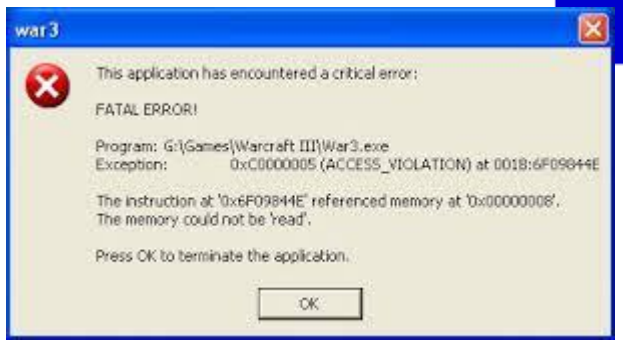
JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

Fatal Exception



PART 12

Exception Handling

What is an Exception

```
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;

        System.out.print("Please enter the divisor\n--> ");
        b = keyboard.nextInt();
        System.out.println("a/b result is : "+(double)a/b);
}
```

```
run:
Please enter the divisor
--> 2
a/b result is : 5.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

What is an Exception

```
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;

        System.out.print("Please enter the divisor\n--> ");
        b = keyboard.nextInt();
        System.out.println("a/b result is : "+(double)a/b);
}
```

```
run:
Please enter the divisor
--> 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at course4.Course4.main(Course4.java:21)
Java Result: 1
BUILD SUCCESSFUL (total time: 2 seconds)
```

What is an Exception

```java
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;

        System.out.print("Please enter the divisor\n--> ");
        b = keyboard.nextInt();
        System.out.println("a/b result is : "+(double)a/b);
}
```

```
run:
Please enter the divisor
--> two
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at course4.Course4.main(Course4.java:20)
Java Result: 1
BUILD SUCCESSFUL (total time: 7 seconds)
```

What is an Exception

```
public static void main(String[] args) {

        for(int i=0; i<3 ;i++)
            System.out.println("The argument " + i + " is : " + args[i]);
}
```

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at course4.Course4.main(Course4.java:16)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

Exception : How-To

- Using the keyword **try**, we pinpoint the sequence of code which may lead to an error

```
try
{
//  the line(s) which may contain an error
}
catch(ArithmeticException e)
{
// the treatment of the error if there is actually an error
}
```

- Using the keyword **catch**, we define the piece of code to run if there is actually an error

Exception : How-To

```
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;

        System.out.print("Please enter the divisor\n--> ");
        b = keyboard.nextInt();

        try {
            System.out.println("a/b result is : "+a/b);
        }
        catch(ArithmeticException e){
            System.out.println("Alert Exception Found: --> UNEXPECTED DIVISOR");
        }
}
```

ArithmeticException is the type of exception handled in that example
There are several others

Exception : How-To

```
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;

        System.out.print("Please enter the divisor\n--> ");
        b = keyboard.nextInt();

        try {
            System.out.println("a/b result is : "+a/b);
        }
        catch(ArithmeticException e){
            System.out.println("Alert Exception Found: --> UNEXPECTED DIVISOR");
        }
}
```

```
run:
Please enter the divisor
--> 0
Alert Exception Found: --> UNEXPECTED DIVISOR
BUILD SUCCESSFUL (total time: 2 seconds)
```

Exception : Multi-Catch

Syntax of an exception: **multiple catch blocks**

```
try {
    // the line(s) which may contain an error
}
catch (ExceptionType1 e1){
    // the treatment of the error of the type ExceptionType1
    // if there is actually an error
}
catch (ExceptionType2 e2){
    // the treatment of the error of the type ExceptionType2
    // if there is actually an error
}
```

Pay attention to put catch blocks **from the more specific to the more generic**, in order to handle the exception as precisely as possible

Exception : Multi-Catch

```
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;

        System.out.print("Please enter the divisor\n--> ");
        try {
            b = keyboard.nextInt();
            System.out.println("a/b result is : "+a/b);
        }
        catch(ArithmeticException e){
            System.out.println("Alert Exception Found: --> UNEXPECTED DIVISOR");
        }
        catch(InputMismatchException e){
            System.out.println("Alert Exception Found: --> UNEXPECTED INPUT");
        }
}
```

```
run:
Please enter the divisor
--> 2.0
Alert Exception Found: --> UNEXPECTED INPUT
BUILD SUCCESSFUL (total time: 3 seconds)
```

Exception : Multi-Catch

```java
public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        int a = 10;
        int b;
        String arg;

        System.out.print("Please enter the divisor\n--> ");
        try {
            b = keyboard.nextInt();
            System.out.println("a/b result is : "+a/b);

            arg = args[0];
        }
        catch(ArithmeticException e){
            System.out.println("Alert Exception Found: --> UNEXPECTED DIVISOR");
        }
        catch(InputMismatchException e){
            System.out.println("Alert Exception Found: --> UNEXPECTED INPUT");
        }
        catch(Exception e){
            System.out.println("Alert Exception Found: --> UNKNOWN REASON");
        }
}
```

```
run:
Please enter the divisor
--> 2
a/b result is : 5
Alert Exception Found: --> UNKNOWN REASON
BUILD SUCCESSFUL (total time: 5 seconds)
```

Exception : Finally

Syntax of an exception: **finally block of code**

```
try {
    // the line(s) which may contain an error
}
catch (ExceptionType1 e1){
    // the treatment of the error of the type ExceptionType1
    // if there is actually an error
}
catch (ExceptionType2 e2){
    // the treatment of the error of the type ExceptionType2
    // if there is actually an error
}
finally {
    // what needs to be done in any case
}
```

Used to ensure that some code is always done (files will be closed at the end of the program for example)

Exception and methods

Sometimes it is useful to manage an exception not in a method but inside the code which has been used to call the exception

```
public static void main(String[] args){
    int i=0, j=0;

    try {
        divide (i, j);
    }
    catch (DivisionByZero e){
        System.out.println(e.getMessage());
    }
}
public static void divide (int i, int j) throws DivisionByZero {
    if (j==0)
        throw new DivisionByZero("Incorrect divisor value");
    System.out.println("Division os correct : " + i + "/" + j + " = " + i/j);
}
```

```java
/**
 * @author Mikael.morelle
 */
public class Player {
    private String nickName;
    private Integer score;

    public Player(String nickName, int score){
        this.nickName = nickName;
        this.score = score;
    }

    public Player(String nickName){
        this.nickName = nickName;
        this.score = null;
    }

    public void setLastScore(int score){
        try{
            if (score < 0 || score > 100)
                throw new Exception("Impossible score value");
            else {
                this.score = score;
                System.out.println("score registered");
            }
         }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    }

    public void setNickName(String nickName) throws Exception {
        try {
           if((nickName == null) || (nickName.isEmpty()))
                    throw new MyException("Invalid nickName");
              else
                    this.nickName = nickName;
       }
       finally{
            System.out.println("Inside Finally Block");
       }

       System.out.println("Outside Finally Block");
    }
}
```

```java
public static void main(String[] args) {

    Player test = new Player("Einstein");

    test.setLastScore(-10);

    test.setLastScore(150000);

    test.setLastScore(75);
}
```

```
run:
Impossible score value
Impossible score value
score registered
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
/**
 * @author Mikael.morelle
 */
public class Player {
    private String nickName;
    private Integer score;

    public Player(String nickName, int score){
        this.nickName = nickName;
        this.score = score;
    }

    public Player(String nickName){
        this.nickName = nickName;
        this.score = null;
    }

    public void setLastScore(int score){
        try{
            if (score < 0 || score > 100)
                throw new Exception("Impossible score value");
            else {
                this.score = score;
                System.out.println("score registered");
            }
         }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    }

    public void setNickName(String nickName) throws Exception {
        try {
           if((nickName == null) || (nickName.isEmpty()))
                    throw new MyException("Invalid nickName");
              else
                    this.nickName = nickName;
       }
       finally{
            System.out.println("Inside Finally Block");
       }

       System.out.println("Outside Finally Block");
    }
}
```

```java
public static void main(String[] args) {

    Player test = new Player("Einstein");

    test.setLastScore(75);
    try {
        test.setNickName(null);
        System.out.println("nickName test passed");
    }
    catch(Exception gotIt){
        System.err.println(gotIt.getMessage());
    }
}
```

```
run:
score registered
Invalid nickName
Inside Finally Block
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
/**
 * @author Mikael.morelle
 */
public class Player {
    private String nickName;
    private Integer score;

    public Player(String nickName, int score){
        this.nickName = nickName;
        this.score = score;
    }

    public Player(String nickName){
        this.nickName = nickName;
        this.score = null;
    }

    public void setLastScore(int score){
        try{
            if (score < 0 || score > 100)
                throw new Exception("Impossible score value");
            else {
                this.score = score;
                System.out.println("score registered");
            }
         }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    }

    public void setNickName(String nickName) throws Exception {
        try {
           if((nickName == null) || (nickName.isEmpty()))
                    throw new MyException("Invalid nickName");
              else
                    this.nickName = nickName;
       }
       finally{
            System.out.println("Inside Finally Block");
       }

       System.out.println("Outside Finally Block");
    }
}
```

```java
public static void main(String[] args) {

    Player test = new Player("Einstein");

    test.setLastScore(75);
    try {
        test.setNickName("");
        System.out.println("nickName test passed");
    }
    catch(Exception gotIt){
        System.err.println(gotIt.getMessage());
    }
}
```

```
run:
score registered
Invalid nickName
Inside Finally Block
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
/**
 * @author Mikael.morelle
 */
public class Player {
    private String nickName;
    private Integer score;

    public Player(String nickName, int score){
        this.nickName = nickName;
        this.score = score;
    }

    public Player(String nickName){
        this.nickName = nickName;
        this.score = null;
    }

    public void setLastScore(int score){
        try{
            if (score < 0 || score > 100)
                throw new Exception("Impossible score value");
            else {
                this.score = score;
                System.out.println("score registered");
            }
         }
        catch(Exception e){
            System.err.println(e.getMessage());
        }
    }

    public void setNickName(String nickName) throws Exception {
        try {
           if((nickName == null) || (nickName.isEmpty()))
                    throw new MyException("Invalid nickName");
              else
                    this.nickName = nickName;
       }
       finally{
            System.out.println("Inside Finally Block");
       }

       System.out.println("Outside Finally Block");
    }
}
```

```java
public static void main(String[] args) {

    Player test = new Player("Einstein");

    test.setLastScore(75);
    try {
        test.setNickName("correctNickName");
        System.out.println("nickName test passed");
    }
    catch(Exception gotIt){
        System.err.println(gotIt.getMessage());
    }
}
```

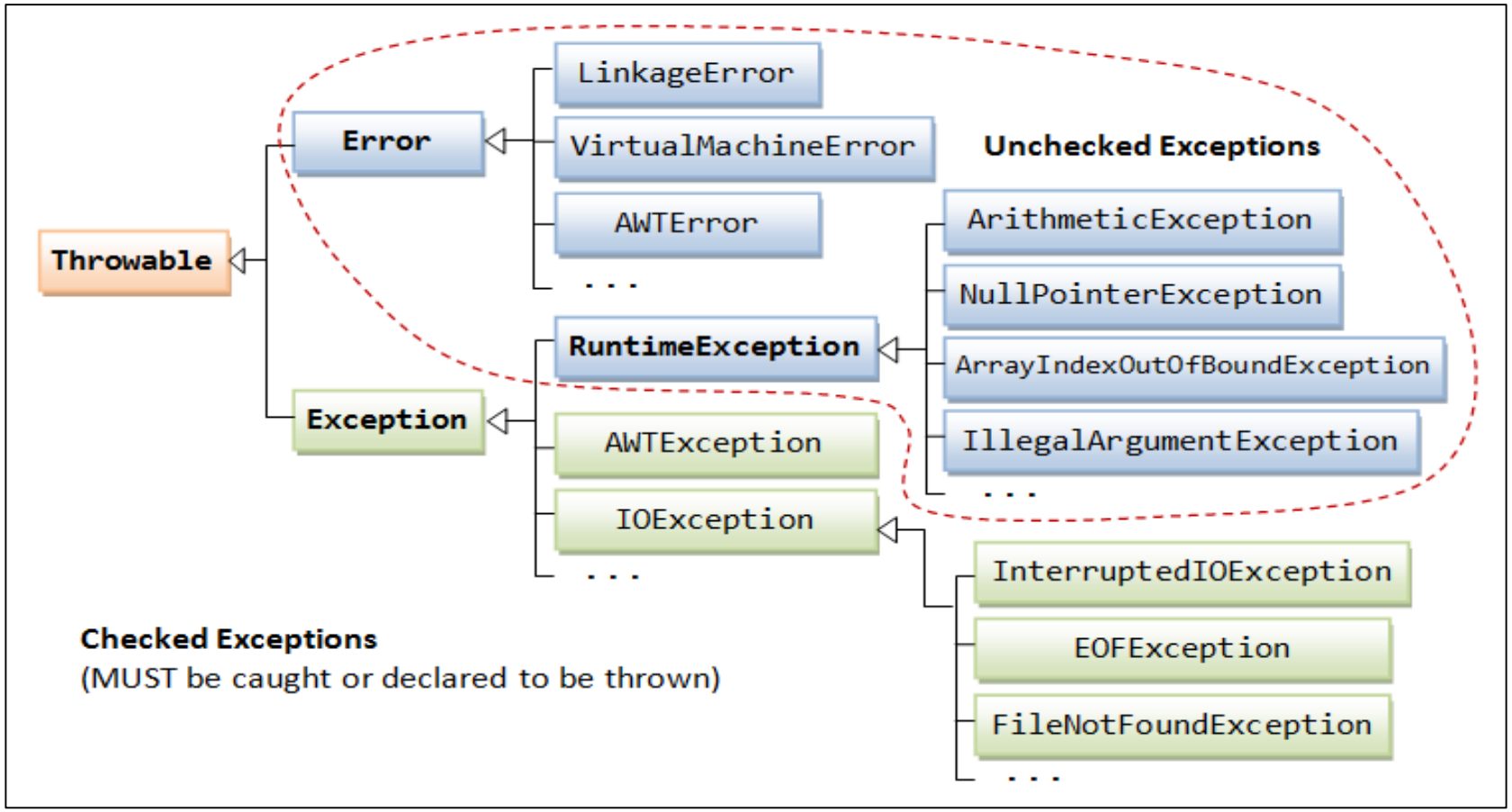
```
run:
score registered
Inside Finally Block
Outside Finally Block
nickName test passed
BUILD SUCCESSFUL (total time: 0 seconds)
```

Predefined Exceptions

- There are some exceptions already defined inside the java.lang package.
- Those are generally subclasses of the standard type RunTimeException.

- There are two kinds of exceptions:
  - **Checked**
  - **UnChecked**

Exceptions


Throwable
Error
LinkageError
VirtualMachineError
AWTError
...
Exception
RuntimeException
ArithmeticException
NullPointerException
ArrayIndexOutOfBoundException
IllegalArgumentException
...
AWTException
IOException
...
InterruptedIOException
EOFException
FileNotFoundException
...
Unchecked Exceptions
Checked Exceptions
(MUST be caught or declared to be thrown)

Predefined Exceptions

Checked Exception: compile time errors regarding class not found and interface not found. It happens during the compilation.

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

Predefined Exceptions

Unchecked exceptions : programmatic runtime errors, due to an error from the user input.

It happens during the program execution.

Exception	Description
ArithmeticException	Arithmetic error, such as divide-by-zero.
ClassCastException	Invalid cast.
ArrayStoreException	Assignment to an array element of an incompatible type.
NullPointerException	Invalid use of a null reference.
IllegalArgumentException	Illegal argument used to invoke a method.
NegativeArraySizeException	Array created with a negative size.

Creating your own Exceptions

```java
public class MyException extends Exception{

    public MyException (String message){
        super(message);
    }

    @Override
    public String getMessage() {
        return "Personalized message : " + super.getMessage() + " !!!";
    }
}
```

```
Debugger Console | Course4 (run)

run:
score registered
Inside Finally Block
Personalized message : Invalid nickName !!!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Multiple Exceptions in 1

```
    public static void main(String[] args) {

        Scanner keyboard = new Scanner (System.in);
        int i = 10;
        int j, result;

        try {
            System.out.print("Enter a number --> ");
            j=keyboard.nextInt();
            result = i/j;

            System.out.println("Division " + i + "/" + j + " = " + result);
        }
        catch(ArithmeticException | InputMismatchException  gotIt){
            System.err.println(gotIt.getMessage());
        }
    }
}
```

```
ourse4.Course4
t
ebugger Console | Course4 (run)
run:
Enter a number --> 0
/ by zero
BUILD SUCCESSFUL (total time: 3 seconds)
```

Multiple Exceptions in 1

```
    public static void main(String[] args) {

        Scanner keyboard = new Scanner (System.in);
        int i = 10;
        int j, result;

        try {
            System.out.print("Enter a number --> ");
            j=keyboard.nextInt();
            result = i/j;

            System.out.println("Division " + i + "/" + j + " = " + result);
        }
        catch(ArithmeticException | InputMismatchException  gotIt){
            System.err.println(gotIt.getMessage());
        }
    }
}
```

```
ourse4.Course4
t
ebugger Console | Course4 (run)
run:
Enter a number --> two
null
BUILD SUCCESSFUL (total time: 5 seconds)
```

PART 13

Files Management

Creating, opening and closing a file
Reading and writing data in a file

FILE Introduction

- Package java.io

- Every operation of Input/Output in java follows these steps:

  1. Opening the stream
  2. Reading or writing data
  3. Closing the stream

an Input Stream is a data sequence read from a source

an Output Stream is a data sequence sent to a destination

File Class

You can use an object of File data type

default path = relative path
→ project directory

Allows to get and set information about the File

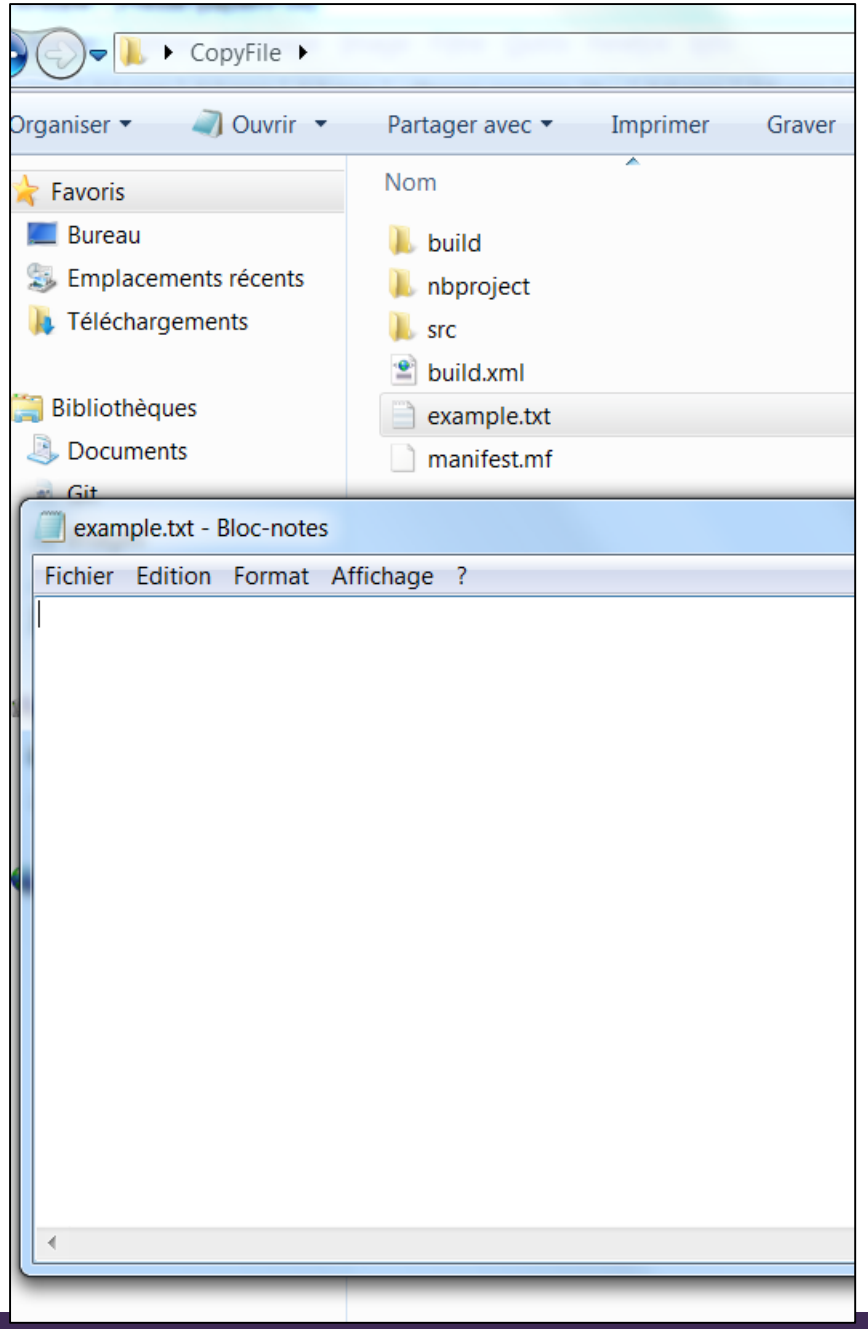
Size

Name

Path

Last Modified

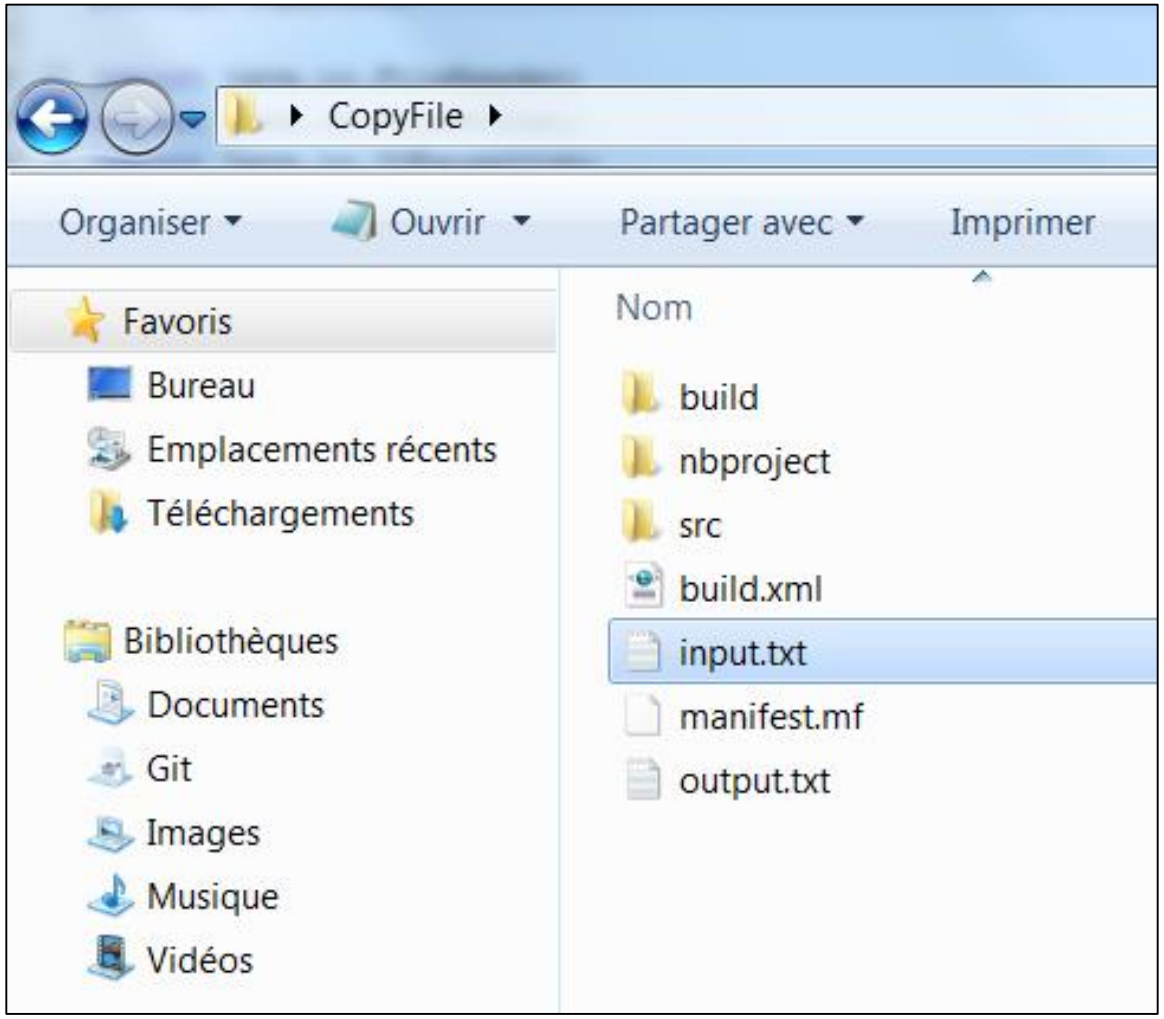
…



File Class

```
import java.io.File;
import java.io.IOException;


public class FileCreation {
	public static void main(String[] args) throws IOException{

		File myFile = new File("example.txt");

		if(myFile.createNewFile())
			System.out.println("File has been correctly created");
		else
			System.out.println("Something wrong happened");
	}
}
```

File location

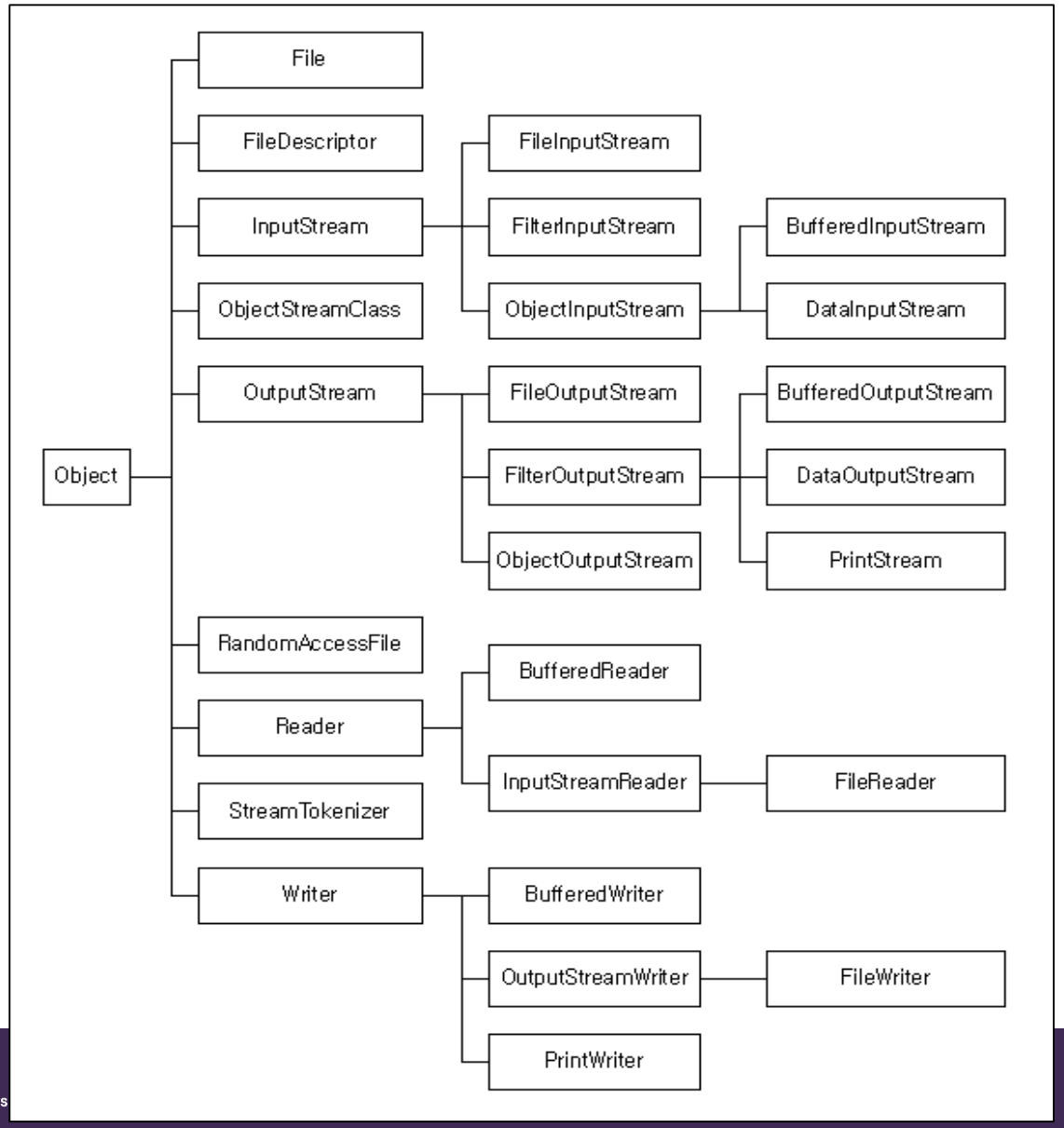


File reading - writing

```java
public class CopyFile2 {

    public static void main(String[] args) throws IOException{
        FileReader in = null;
        FileWriter out = null;

        try {
            in = new FileReader("C:\\JavaFileSamples\\input.txt");
            out = new FileWriter("C:\\JavaFileSamples\\output.txt");

            int c;
            while((c = in.read()) != -1) {
                out.write(c);
            }
        }
        finally {
            if(in != null)in.close();
            if(out != null)out.close();
        }
    }
}
```

You can define a specific location by providing an exact location → Absolute path

Files Management



Binary Mode
1 Byte stream

Input = Reading

Output = Writing

Text Mode
2 Bytes stream

JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

1 Byte sample

```java
package copyfile;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/**
 *
 * @author Mikael.Morelle
 */
public class CopyFile1 {


    public static void main(String[] args) throws IOException{
        FileInputStream in = null;
        FileOutputStream out = null;

        try {
            in = new FileInputStream("input.txt");
            out = new FileOutputStream("output.txt");

            int c;
            while((c = in.read()) != -1) {
                out.write(c);
            }
        }
        finally {
            if(in != null)in.close();
            if(out != null)out.close();
        }
    }
}
```

```
ut
Debugger Console | Course4 (run) | CopyFile (run)
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

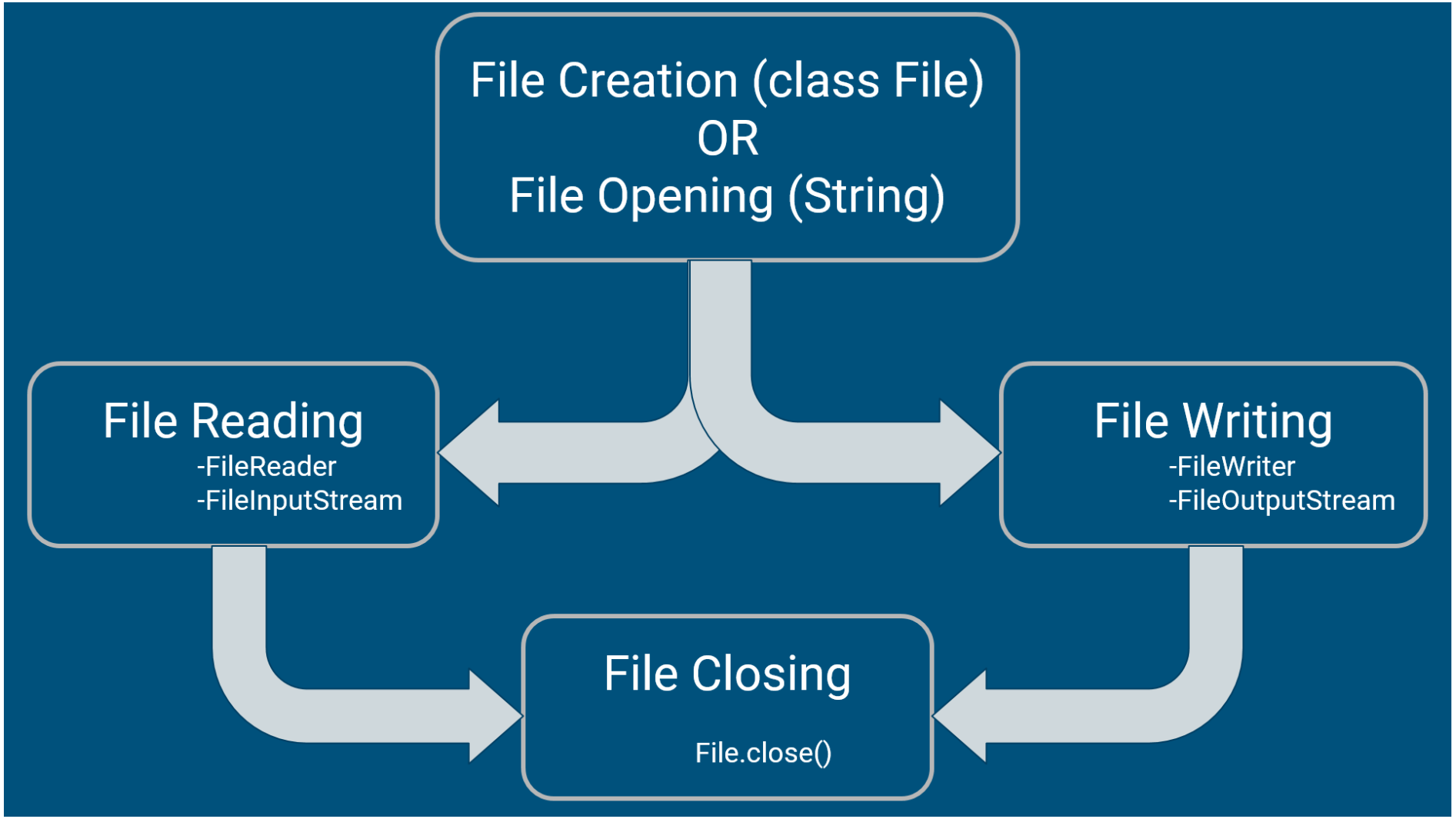
JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

Character sample

```
package copyfile;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

/**
 *
 * @author Mikael.Morelle
 */
public class CopyFile2 {

    public static void main(String[] args) throws IOException{
        FileReader in = null;
        FileWriter out = null;

        try {
            in = new FileReader("input.txt");
            out = new FileWriter("output.txt");

            int c;
            while((c = in.read()) != -1) {
                out.write(c);
            }
        }
        finally {
            if(in != null)in.close();
            if(out != null)out.close();
        }
    }
}
```

```
ebugger Console | CopyFile (run)
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

File Summary



File Reading

FileInputStream: used for reading data bytes from files

```
InputStream myFile = new FileInputStream("C:\\java\\hello");
    OR
File myFile = new File("C:\\java\\hello");
InputStream file = new FileInputStream(myFile);
```

FileReader: used for reading data characters from files

```
FileReader myFile = new FileReader("C:\\java\\hello");
    OR
File myFile = new File("C:\\java\\hello");
FileReader file = new FileReader(myFile);
```

File Writing

FileOutputStream: used for writing data bytes into file

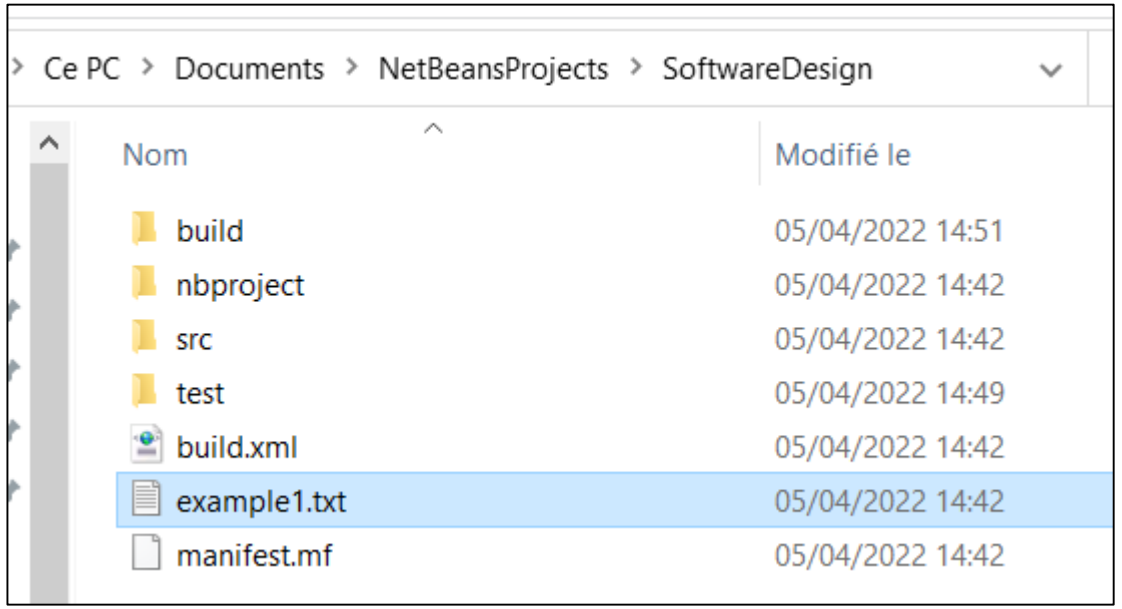
```
OutputStream myFile = new FileOutputStream("C:\\java\\hello");
    OR
File myFile = new File("C:\\java\\hello");
OutputStream file = new FileOutputStream(myFile);
```

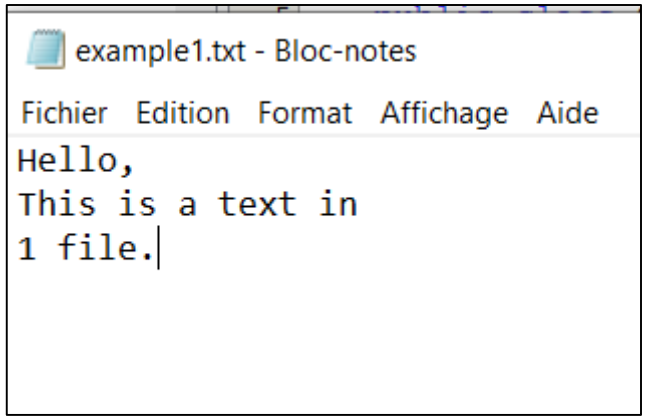
FileWriter: used for writing characters into file

```
FileWriter myFile = new FileWriter("C:\\java\\hello");
    OR
File myFile = new File("C:\\java\\hello");
FileWriter file = new FileWriter(myFile);
```

File example

```java
package softwaredesign;
import java.io.File;

public class SoftwareDesign {

    public static void main(String[] args) {
        File file = new File("./example1.txt");
    }
}
```





File example

```
package softwaredesign;

import java.io.File;
import java.util.Scanner;


public class SoftwareDesign {

    public static void main(String[] args) {

        File file = new File("./example1.tx
        Scanner lecturer = new Scanner(file);
    }

}
```

unreported exception FileNotFoundException; must be caught or declared to be thrown

Explict type can be replaced with 'var'

(Alt-Enter shows hints)

File example

```
package softwaredesign;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class SoftwareDesign {
    public static void main(String[] args) {

        File file = new File("./example1.txt");
        try{
            Scanner lecturer = new Scanner(file);
        }
        catch (FileNotFoundException f){
            System.out.println(f.getMessage());
        }
    }
}
```

File example

```java
package softwaredesign;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class SoftwareDesign {
    public static void main(String[] args) {
        String text;
        File file = new File("./example1.txt");
        try{
            Scanner lecturer = new Scanner(file);
            do{
                text = lecturer.nextLine();
                System.out.println(text);
            }
            while(lecturer.hasNext());
        }
        catch (FileNotFoundException f){
            System.out.println(f.getMessage());
        }
    }
}
```

```
compile:
run:
Hello,
This is a text in
1 file.
BUILD SUCCESSFUL (total time: 2 seconds)
```