

Exception Handling

Christian Rodríguez Bustos

Edited by Juan Mendivelso

Object Oriented Programming



Agenda



CPUID: GenuineIntel 5.2.c irq1:1f SYSVER 0xf0000565

Dll	Base	DateStmp	Name	Dll	Base	DateStmp	Name
80100000	3202c07e		ntoskrnl.exe	80010000	31ee6c52		hal.dll
80001000	31ed06b4		atapi.sys	80006000	31ec6c74		SCSI\PORT.SYS
802c6000	31ed06bf		aic78xx.sys	802cd000	31ed237c		Disk.sys
802d1000	31ec6c7a		CLASS2.SYS	8037c000	31eed0a7		Ntfs.sys
fc698000	31ec6c7d		Floppy.SYS	fc6a8000	31ec6ca1		Cdrom.SYS
fc90a000	31ec6df7		Fs_Rec.SYS	fc9c9000	31ec6c99		Null.SYS
fc864000	31ed868b		KSecDD.SYS	fc9ca000	31ec6c78		Beep.SYS
fc6d8000	31ec6c98		i8042prt.sys	fc86c000	31ec6c97		mouclass.sys
fc874000	31ec6c94		kbdclass.sys	fc6f0000	31f50722		VIDEO\PORT.SYS
feffa000	31ec6c62		mga_mil.sys	fc890000	31ec6c6d		vga.sys
fc708000	31ec6ccb		Msfs.SYS	fc4b0000	31ec6cc7		Npfs.SYS
feebc000	31eed262		NDIS.SYS	a0000000	31f954f2		win32k.sys
feba4000	31f						.SYS
feb8c000	31f						.SYS
feacf000	31f						.SYS
fc350000	31f						.SYS
fc710000	31f						.SYS
fc870000	31f						.SYS
fc5b0000	31f						.SYS
fea3b000	31f						.SYS

Address dwo
fec32d84 801
801471c8 801
801471dc 801
80147304 803

Restart and set the recovery options in the system control panel or the /CRASHDEBUG system start option.

Application Error



Bluescreen has performed an illegal operation. Bluescreen must be closed.

OK

YOU FAIL AT FAILING

No, that's not a double negative.

DIYDESPAIR.COM

No matter how good developer you are, you can not control everything.

CPUID: GenuineIntel 5.2.c irq1:1f SYSVER 0xf0000565

Dll	Base	DateStmp	Name	Dll	Base	DateStmp	Name
80100000	3202c07e		ntoskrnl.exe	80010000	31ee6c52		hal.dll
80001000	31ed06b4		atapi.sys	80006000	31ec6c74		SCSI\PORT.SYS
802c6000	31ed06bf		aic78xx.sys	802cd000	31ed237c		Disk.sys
802d1000	31ec6c7a		CLASS2.SYS	8037c000	31eed0a7		Ntfs.sys
fc698000	31ec6c7d		Floppy.SYS	fc6a8000	31ec6ca1		Cdrom.SYS
fc90a000	31ec6df7		Fs_Rec.SYS	fc9c9000	31ec6c99		Null.SYS
fc864000	31ed868b		KSecDD.SYS	fc9ca000	31ec6c78		Beep.SYS
fc6d8000	31ec6c98		i8042prt.sys	fc86c000	31ec6c97		mouclass.sys
fc874000	31ec6c94		kbdclass.sys	fc6f0000	31f50722		VIDEO\PORT.SYS
feffa000	31ec6c62		rga_mil.sys	fc890000	31ec6c6d		vga.sys
fc708000	31ec6ccb		Msfs.SYS	fc4b0000	31ec6cc7		Npfs.SYS
fefbc000	31eed262		NDIS.SYS	a0000000	31f954f2		win32k.sys
feba4000	31f						.SYS
feb8c000	31f						.SYS
feacf000	31f						.SYS
fc550000	31f						.SYS
fc710000	31f						.SYS
fc870000	31f						.SYS
fc5b0000	31f						.SYS
fea3b000	31f						.SYS

Address dwo
fec32d84 801
801471c8 801
801471dc 801
80147304 803

Restart and set the recovery options in the system control panel or the /CRASHDEBUG system start option.

YOU FAIL AT FAILING

No, that's not a double negative.

DIYDESPAIR.COM

Your code must be prepared to *handle exceptional situations*

1. Exception Handling Basis

1.1 Exceptions

1.2 Try and Catch

1.3 Finally

1.1 Exceptions

Exceptions are Java run-time errors

Run-time errors (**exceptions**) are common if we do not use **exception handling** to deal with them

Out of bounds array access

Operate null objects

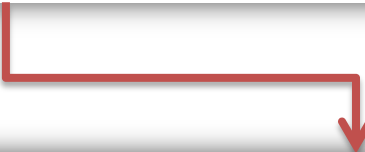
Division by zero

Invalid casting

Exceptions are Java run-time errors

```
// ...  
private static void outOfBoundsException() {  
  
    // this code generates  
    // a null pointer exception  
    int array[] = {0, 1, 2, 3, 4};  
  
    System.out.println(array[6]);  
}  
// ...
```

**Out of bounds array
access** throws an
Exception



Output - ExceptionHandling (run)	Search Results	Tasks
<p>run:</p> <p>Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6 at exceptionhandling.ExceptionHandling.outOfBoundsException(ExceptionHandling.java:53) at exceptionhandling.ExceptionHandling.main(ExceptionHandling.java:18)</p> <p>Java Result: 1</p> <p>BUILD SUCCESSFUL (total time: 1 second)</p>		

Exceptions are Java run-time errors

Exception Name

Exception details

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6
    at exceptionhandling.ExceptionHandling.outOfBoundsException(ExceptionHandling.java:53)
    at exceptionhandling.ExceptionHandling.main(ExceptionHandling.java:18)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

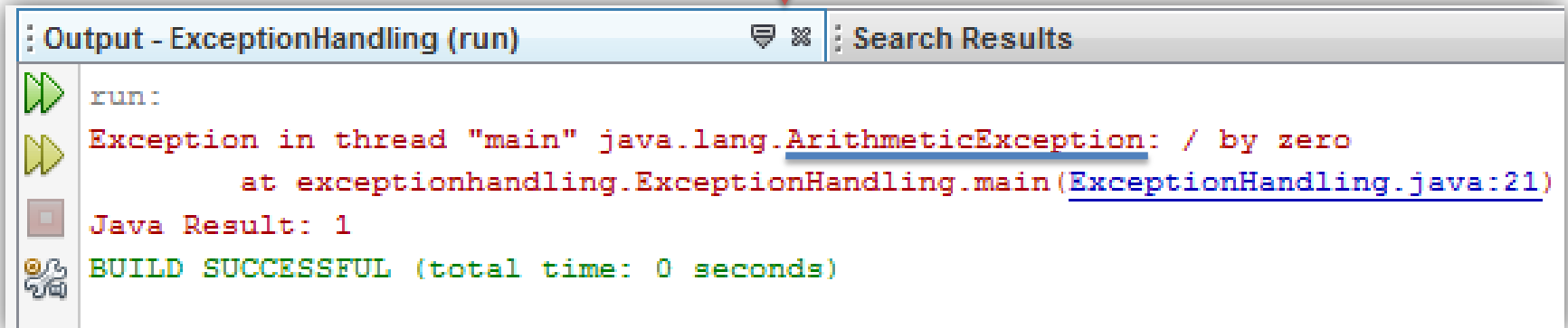
Stack Trace

A user-friendly snapshot of the threads and monitors in a JVM

Exceptions are Java run-time errors

```
// ...  
public static void divisionByZeroException() {  
  
    // this code generates  
    // a division by zero exception  
    int numberA = 5;  
    int numberB = 0;  
  
    int result = numberA / numberB;  
  
    System.out.println(result);  
}  
// ...
```

Division by zero
throws an
Exception (run time
error)



The screenshot shows an IDE's Output window titled "Output - ExceptionHandling (run)". The output text is as follows:


```
run:  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at exceptionhandling.ExceptionHandling.main(ExceptionHandling.java:21)  
Java Result: 1  
BUILD SUCCESSFUL (total time: 0 seconds)
```

A red arrow points from the code block above to the "ExceptionHandling.java:21" link in the output window.

Exceptions are Java run-time errors

**Operate null
objects** throws an
Exception

```
// ...  
private static void nullPointerException() {  
  
    // this code generates  
    // a null pointer exception  
    String name = null;  
  
    name.length();  
  
    System.out.println(name);  
}  
// ...
```




Output - ExceptionHandling (run) Search Results Tasks

run:
Exception in thread "main" java.lang.NullPointerException
 at exceptionhandling.ExceptionHandling.nullPointerException(ExceptionHandling.java:36)
 at exceptionhandling.ExceptionHandling.main(ExceptionHandling.java:17)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)

Exceptions are Java run-time errors

```
// ...  
private static void castingException() {  
  
    Scanner reader = new Scanner(System.in);  
  
    System.out.println("Please, select a number: ");  
    int userOption = reader.nextInt();  
}  
// ...
```

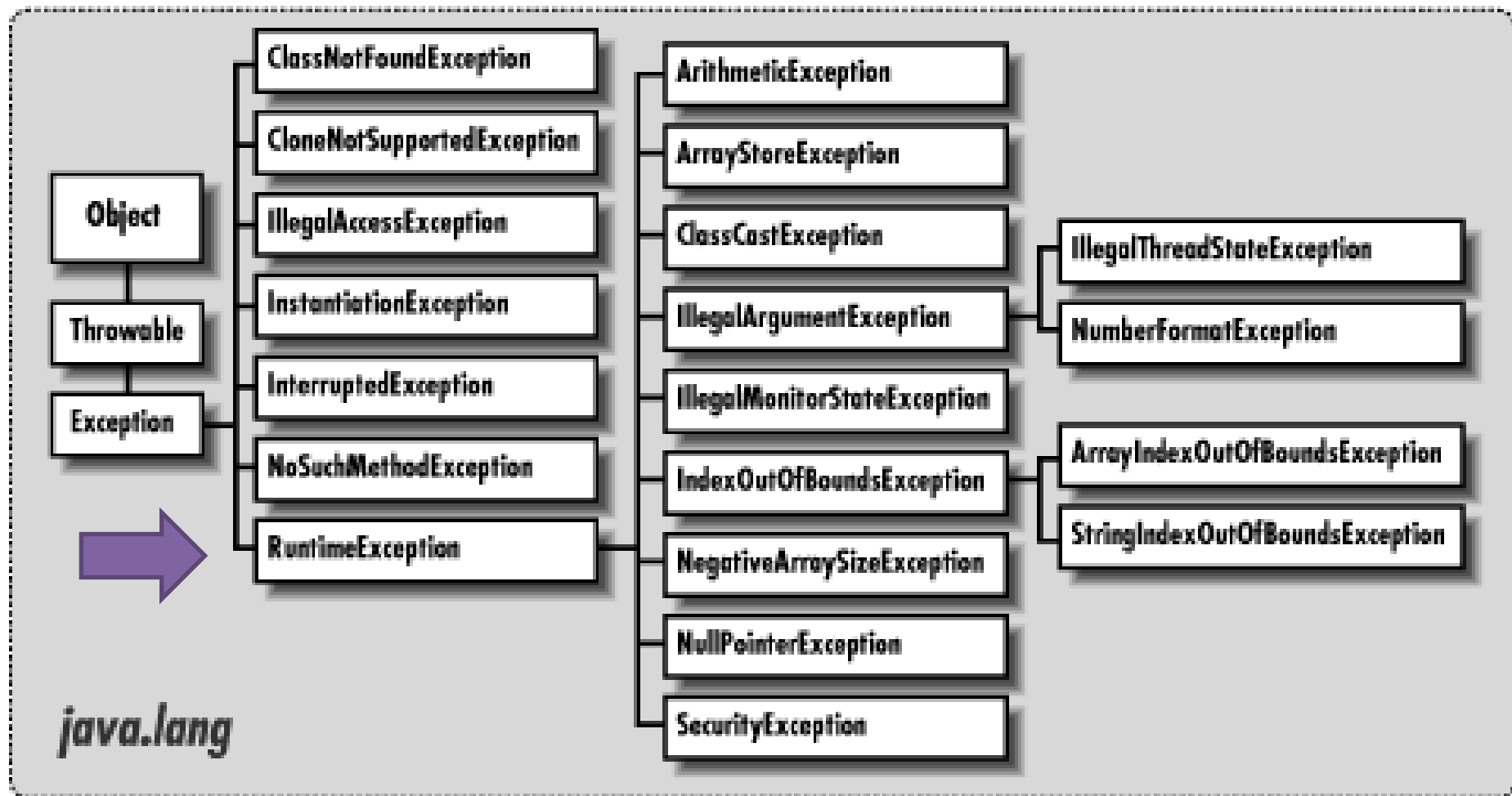
Invalid casting
throws an
Exception



Output - ExceptionHandling (run) Search Results

```
run:  
Please, select a number:  
Five  
Exception in thread "main" java.util.InputMismatchException  
    at java.util.Scanner.throwFor(Scanner.java:840)  
    at java.util.Scanner.next(Scanner.java:1461)  
    at java.util.Scanner.nextInt(Scanner.java:2091)  
    at java.util.Scanner.nextInt(Scanner.java:2050)  
    at exceptionhandling.ExceptionHandling.castingException(ExceptionHandling.java:63)  
    at exceptionhandling.ExceptionHandling.main(ExceptionHandling.java:20)  
Java Result: 1  
BUILD SUCCESSFUL (total time: 10 seconds)
```

The compiler checks for everything except RuntimeExceptions.



http://docstore.mik.ua/oreilly/java/langref/ch09_04.htm

Exception Handling is “Plan B” to handle errors

Exception Handling allows us to be prepared for **unexpected run-time errors**



1.2 Try and Catch

Try and catch are used to handle exceptions

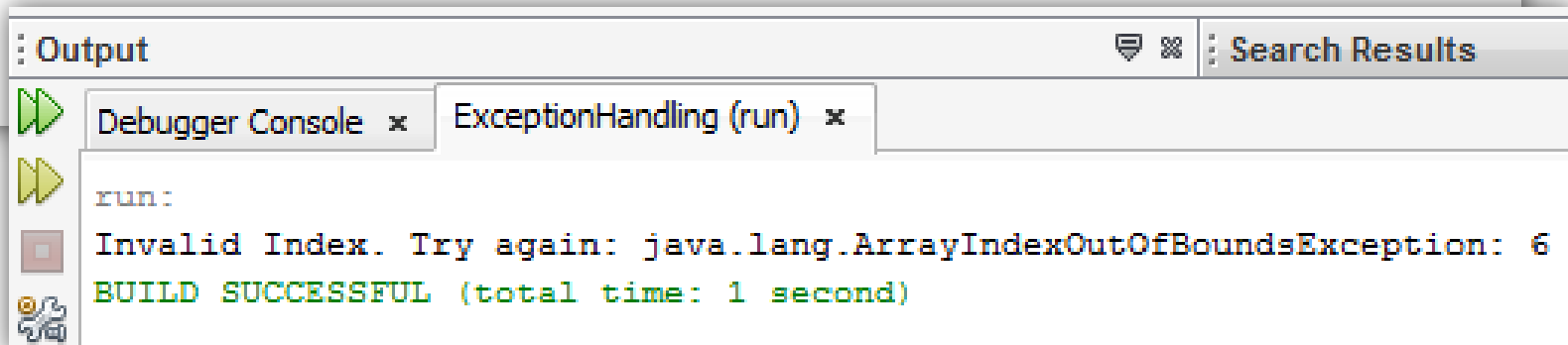
```
// ...  
private static void TryAndCatch() {  
    → try {  
        // To do some risky things and  
    } catch (Exception exception) {  
        // If something exceptional happens  
        // execute the plan B (try to recover)  
        // This code only runs if an Exception  
        // is thrown  
    }  
}  
// ...
```

Java *Try and Catch*
structure example

Try and catch are used to handled exceptions

```
// ...
private static void outOfBoundsException() {
    try {
        int array[] = {0, 1, 2, 3, 4};
        System.out.println(array[6]);
    } catch (IndexOutOfBoundsException exception) {
        System.out.println("Invalid Index. Try again: "+exception);
    }
}
// ...
```

Exception class name



```
Output
Debugger Console x ExceptionHandling (run) x
run:
Invalid Index. Try again: java.lang.ArrayIndexOutOfBoundsException: 6
BUILD SUCCESSFUL (total time: 1 second)
```

Try and catch are used to handled exceptions

```
// ...
public static void divisionByZeroException() {

    try {
        int numberA = 5;
        int numberB = 0;

        int result = numberA / numberB;

        System.out.println(result);

    } catch (ArithmeticException exception) {

        System.out.println("A division by zero has occurred");

    }

}
// ...
```

Exception class name

```
Output
Debugger Console x ExceptionHandling (run) x
run:
A division by zero has occurred
BUILD SUCCESSFUL (total time: 1 second)
```

Try and catch are used to handled exceptions

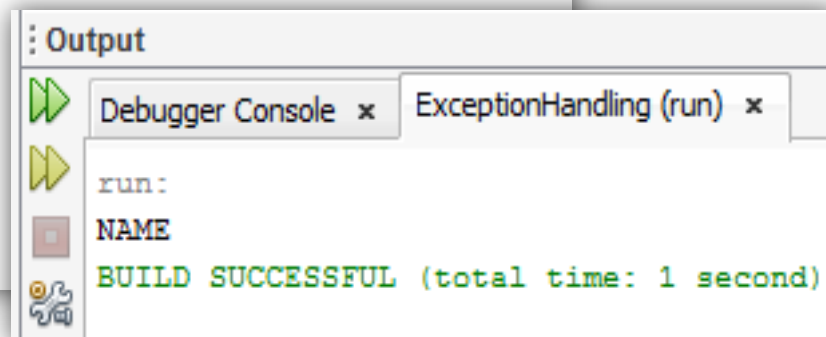
```
// ...
private static void nullPointerException() {

    String name = null;

    try {
        name.length();
    } catch (NullPointerException exception) {
        name = "NAME";
    }

    System.out.println(name);
}
// ...
```

Exception class name



Try and catch are used to handled exceptions

```
// ...
private static void castingException() {

    Scanner reader = new Scanner(System.in);

    System.out.println("Please, select a number: ");

    try {

        int userOption = reader.nextInt();

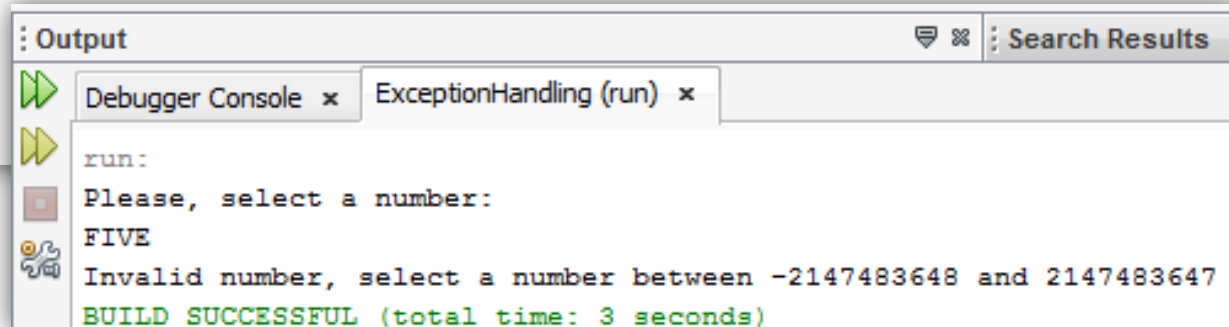
    } catch (InputMismatchException exception) {

        System.out.println("Invalid number, select a number between "
            + Integer.MIN_VALUE + " and " + Integer.MAX_VALUE);

    }

}
// ...
```

Exception class name

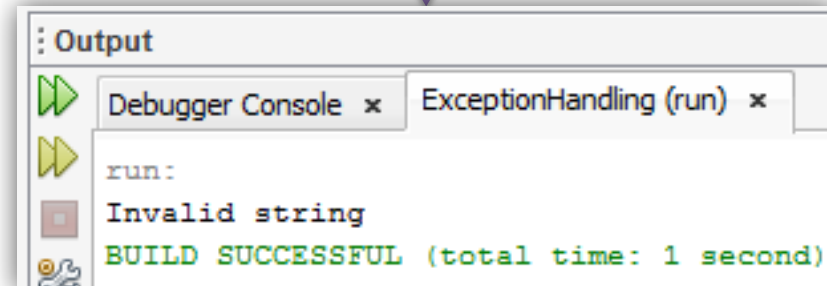


```
Output
Debugger Console x ExceptionHandling (run) x
run:
Please, select a number:
FIVE
Invalid number, select a number between -2147483648 and 2147483647
BUILD SUCCESSFUL (total time: 3 seconds)
```

Try and catch are used to handle exceptions

```
// ...  
private static void castingException() {  
    try {  
        String numberString = "five";  
        int number = Integer.parseInt(numberString);  
    } catch (NumberFormatException ex) {  
        System.out.println("Invalid string");  
    }  
}  
// ...
```


**Invalid casting
throws an
Exception**



1.3 Finally

Finally block is used to do things no matter what happen

```
public static void main(String[] args) throws IOException {  
  
    FileWriter writer = null;  
  
    try {  
  
        writer = new FileWriter("c:\\\\Windows");  
        writer.write("My Test Text file");  
  
    } catch (FileNotFoundException ex) {  
  
        System.out.println("An Error has occurred trying to open the file: ");  
        ex.printStackTrace();  
  
    } catch (IOException ex) {  
  
        ex.printStackTrace();  
  
    } finally {  
  
        System.out.println("File will be released");  
        writer.close();  
  
    }  
  
}
```



This block will be executed whether an exception occurs or not

It is possible to handle to different Exceptions

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

    FileWriter writer = null;

    try {

        writer = new FileWriter("c:\\\\Windows");
        writer.write("My Test Text file");

    } catch (FileNotFoundException ex) {

        System.out.println("An Error has occurred trying to open the file: ");
        ex.printStackTrace();

    } catch (IOException ex) {

        ex.printStackTrace();

    } finally {

        System.out.println("File will be released");
        writer.close();

    }

}
```

1

2

2. Exception Handling Flow

2.1 Who throws exceptions?

2.2 Exception handling flow

2.1 Who throws exceptions?

2.2 Exception handling flow

One method will catch what another method throws

nextInt

```
public int nextInt()
```

Scans the next token of the input as an int.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the int scanned from the input

Throws:

- [InputMismatchException](#) - if the next token does not match the *Integer* regular expression, or is out of range
- [NoSuchElementException](#) - if input is exhausted
- [IllegalStateException](#) - if this scanner is closed

Methods can throw more than one exception



Class Scanner – Method nextInt

One method will catch what another method throws

FileWriter

```
public FileWriter(File file)  
    throws IOException
```

Constructs a FileWriter object given a File object.

Parameters:

file - a File object to write to.

Throws:

IOException - if the file exists but is a directory rather than a regular file, does not exist but cannot be created, or cannot be opened for any other reason

Class FileWriter – Constructor FileWriter(File)

One method will catch what another method throws

exists

```
public boolean exists()
```

Tests whether the file or directory denoted by this abstract pathname exists.

Returns:

true if and only if the file or directory denoted by this abstract pathname exists; false otherwise

Throws:

[SecurityException](#) - If a security manager exists and its [SecurityManager.checkRead\(java.lang.String\)](#) method denies read access to the file or directory

Class File – method exists

Exceptions can be thrown by user defined methods

```
// ...  
private static int castingException() throws InputMismatchException {  
  
    Scanner reader = new Scanner(System.in);  
  
    System.out.println("Please, select a number: ");  
  
    return reader.nextInt();  
  
}  
// ...
```



This Exception will not be handled by this method

Exceptions can be thrown by user defined methods

```
// ...
public static void main(String[] args) {
    try {
        castingException();
    } catch (InputMismatchException ex) {
        System.out.println("Invalid input.");
    }
}

private static int castingException() throws InputMismatchException {

    Scanner reader = new Scanner(System.in);

    System.out.println("Please, select a number: ");

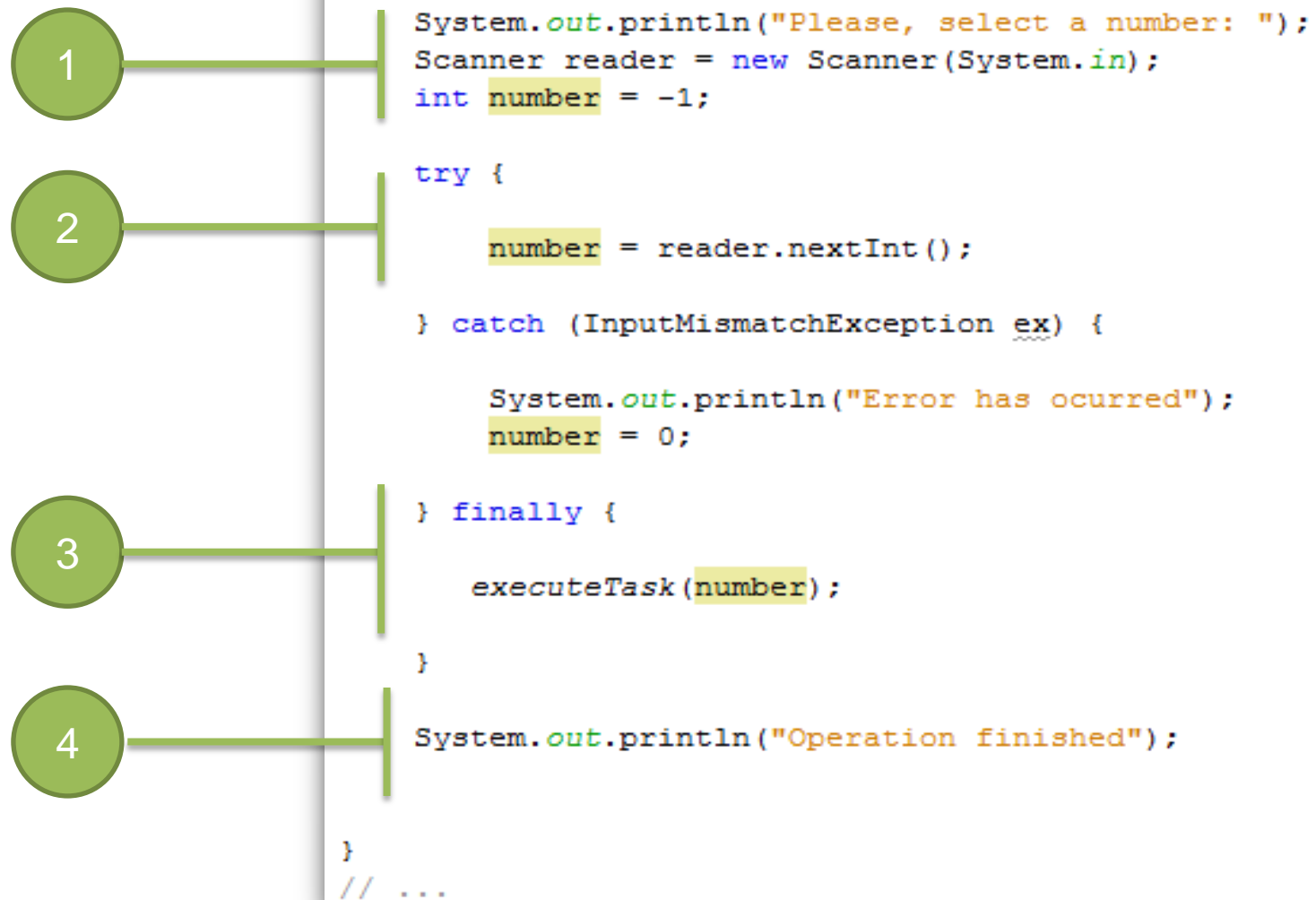
    return reader.nextInt();

}
// ...
```

This Exception will be handled by this method

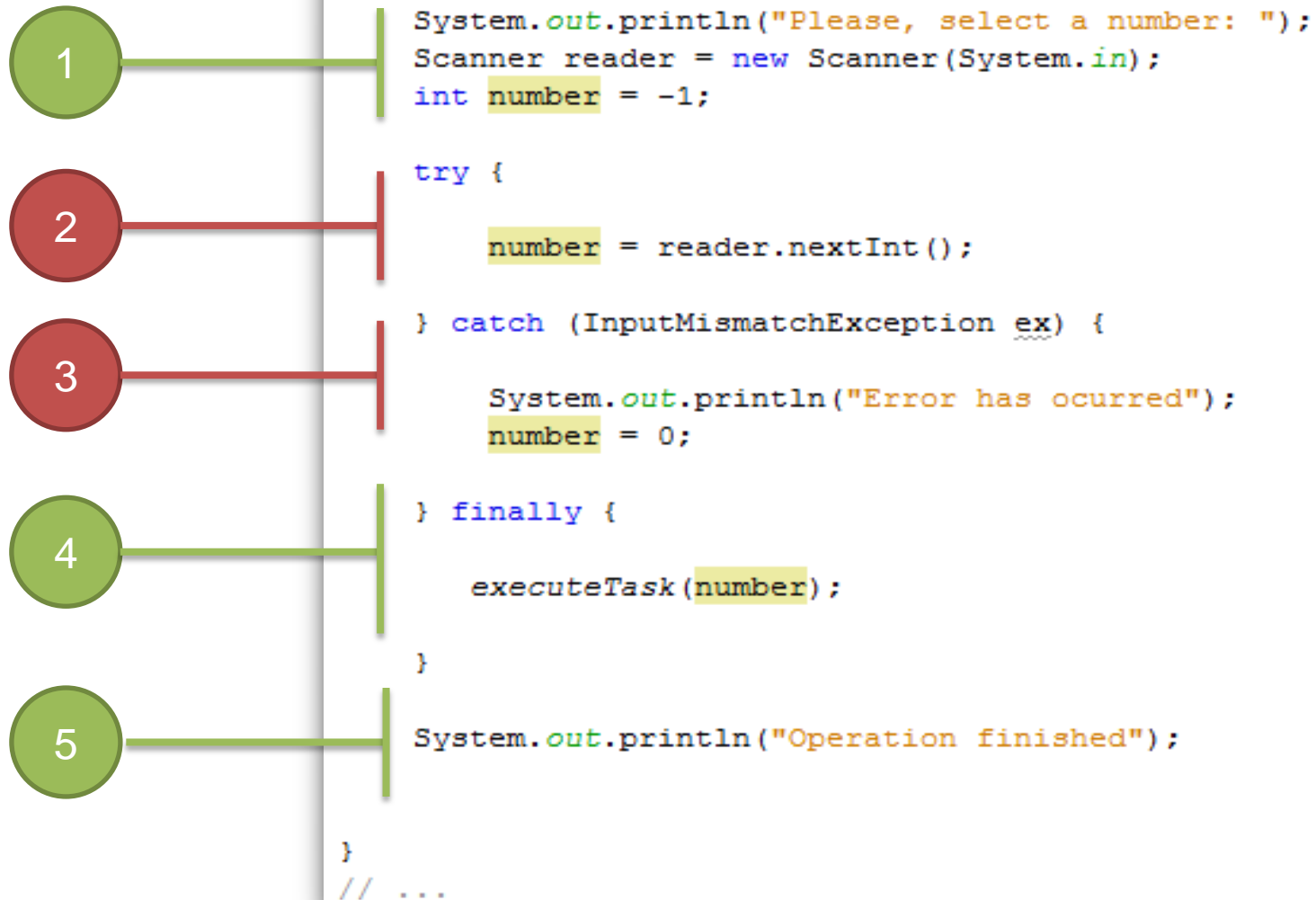
2.2 Exception handling flow

Normal Execution flow




Error Execution flow

Exception thrown



3. Exception Rules


You cannot have a catch or finally without a try



```
// ...
private static void TryAndCatch(
    System.out.println("Try and
catch (Exception exception) {
    }
}
// ...
```

'catch' without 'try'
';' expected

(Alt-Enter shows hints)




```
// ...
private static void TryAndCatch() {
    System.out.println("Try and catch example");
    finally {
    }
}
// ...
```

You cannot put code between the try and the catch

```
// ...
private static void TryAndCatch() {
    Scanner scanner = new Scanner(System.in);

    try {
        int number = scanner.nextInt();

    }
    System.out.println("We can not code here");
    catch (Exception exception) {
        exception.printStackTrace();
    }
}
```


 A red circle with a diagonal slash, indicating that the code between the try and catch blocks is invalid.

'try' without 'catch' or 'finally'
Scan ---- (Alt-Enter shows hints)

A try must be followed by either a catch or a finally

```
// ...  
private static void TryAndCatch() {  
    Scanner scanner = new Scanner(System.in);  
    try {  
        int number = scanner.nextInt();  
    }  
}  
// ...
```

'try' without 'catch' or 'finally'
(Alt-Enter shows hints)



References

[Deitel] H.M. Deitel and P.J. Deitel, *Java How to Program: Early Objects Version*, Prentice Hall, 2009.

[Sierra] K. Sierra and B. Bates, *Head First Java*, 2nd Edition, O'Reilly Media, 2005.