

Programming Software Systems

Lab 9

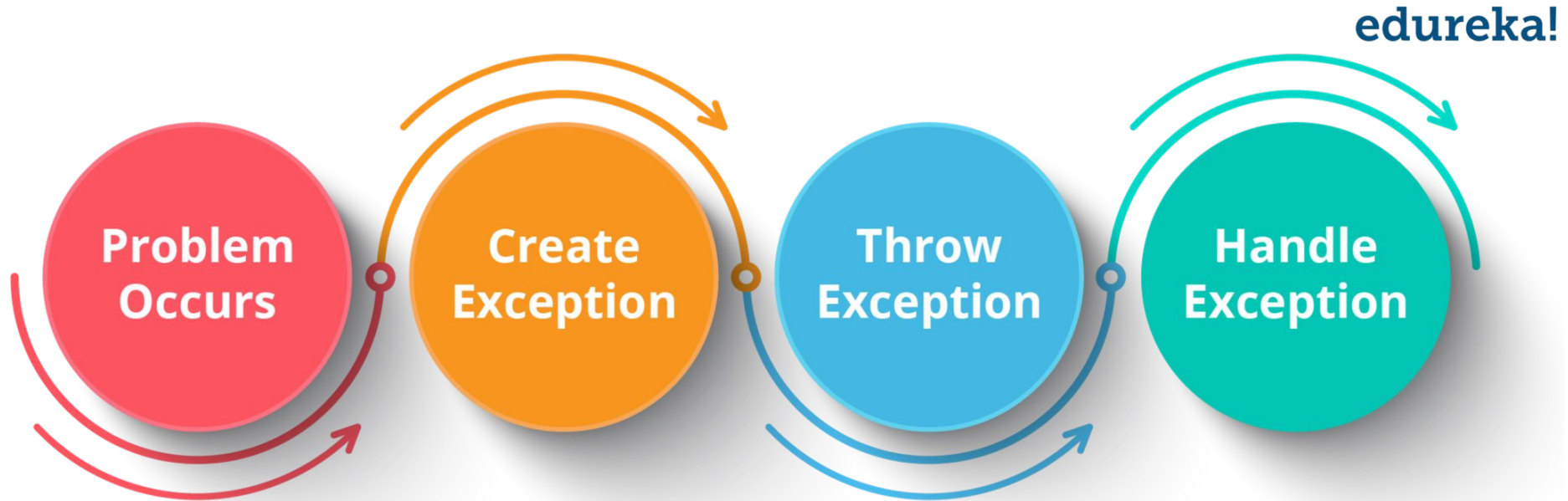
Agenda

- Revision of exceptions in Java: from different angle
- Warm-up exercises
- Solving Problems
- Q&A

Learning outcome:

- Strengthen the error-handling skills in Java
- Application of Error-handling skills in real-world problems
- Code-review & improving code quality

Exception flow



Exceptions in Programming

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Exception categories

We have three categories of Exceptions:

- **Checked exceptions** - an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions
- **Unchecked exceptions**- an exception that occurs at the time of execution. These are also called as *Runtime Exceptions*.
- **Errors** - these are not exceptions, but problems that arise beyond the control of the user or the programmer.

Exceptions...

```
public class Main {  
  
    public static void main(String args[]) {  
        int num[] = {1, 2, 3, 4};  
        System.out.println(num[5]);  
    }  
}
```

Is it **Checked exception**, or **Unchecked exception** or **Error**?

Exceptions...

```
import java.io.File;
import java.io.FileReader;

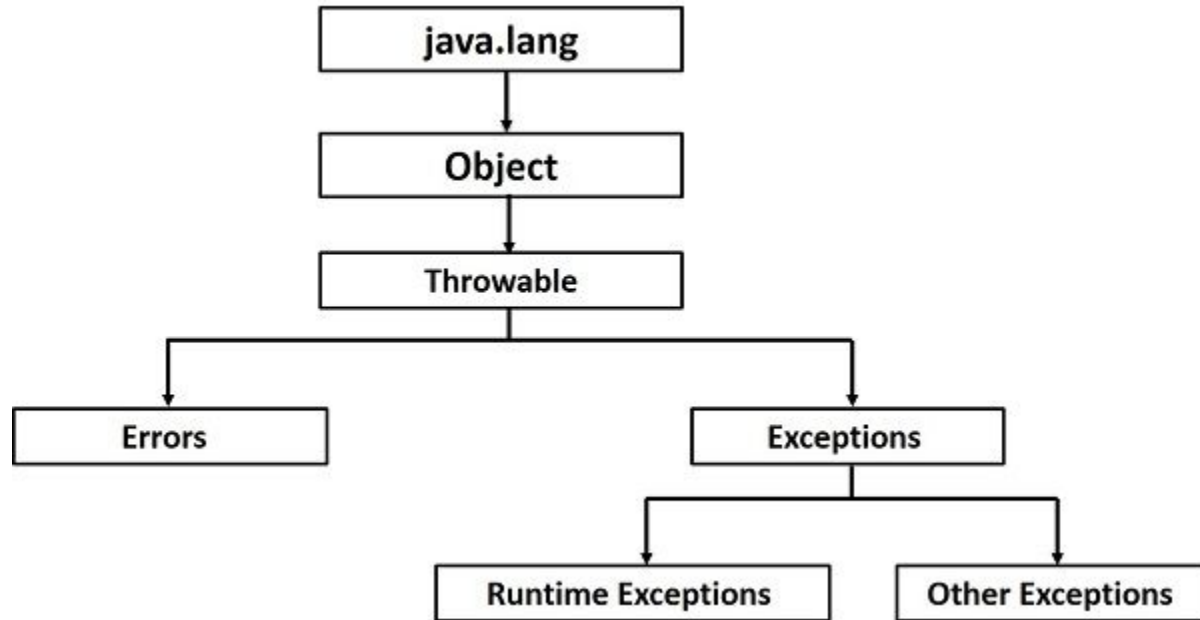
public class Main {
    public static void main(String args[]) {
        File file = new File("E://file.txt");
        FileReader fr = new FileReader(file);
    }
}
```

Is it **Checked exception**, or **Unchecked exception** or **Error**?

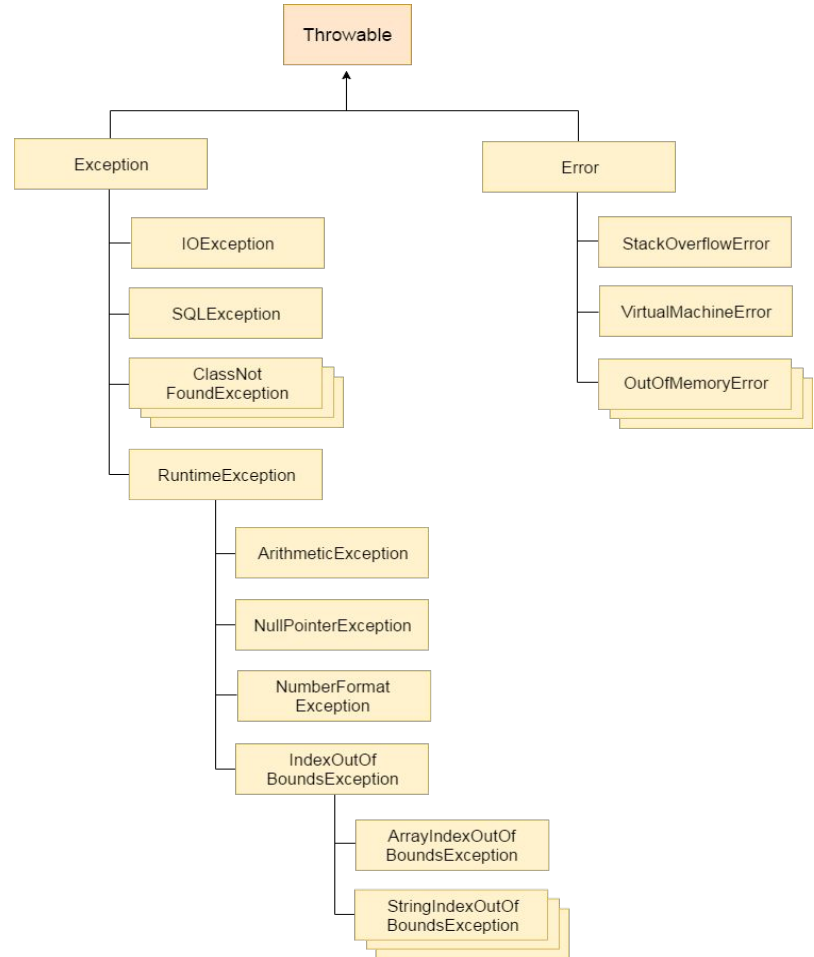
Exceptions...



Exceptions Hierarchy



Exceptions Hierarchy 2




To decrease bugs...

Both good programmers and bad programmers make stupid mistakes. The difference is that good programmers:

- write code that is simpler and easier to debug,
- use tools such as JUnit to help ensure that their code is correct, and
- are not satisfied with code that "mostly" works.

For more details...



The Java™ Tutorials

Hide TOCSearch

Exceptions

« Previous • Trail • Next »

Home Page > Essential Classes

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases and might use technology no longer available. See [Java Language Changes](#) for a summary of updated language features in Java SE 9 and subsequent releases. See [JDK Release Notes](#) for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Lesson: Exceptions

The Java programming language uses *exceptions* to handle errors and other exceptional events. This lesson describes when and how to use exceptions.

What Is an Exception?

An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.

The Catch or Specify Requirement

This section covers how to catch and handle exceptions. The discussion includes the `try`, `catch`, and `finally` blocks, as well as chained exceptions and logging.

How to Throw Exceptions

This section covers the `throw` statement and the `Throwable` class and its subclasses.

The try-with-resources Statement

<https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html>

For more details...

The Java™ Tutorials

Exceptions

- What Is an Exception?
- The Catch or Specify Requirement
- Catching and Handling Exceptions
- The try Block
- The catch Blocks
- The finally Block
- The try-with-resources Statement
- Putting It All Together
- Specifying the Exceptions Thrown by a Method
- How to Throw Exceptions
- Chained Exceptions
- Creating Exception Classes
- Unchecked Exceptions — The Controversy
- Advantages of Exceptions
- Summary
- Questions and Exercises

« Previous • Trail • Next »

The Java Tutorials have been updated for Java SE 8. See Java Language Changes and See JDK Release Notes for details.

Lesson: Exception

The Java programming language

What Is an Exception

An exception is an event that

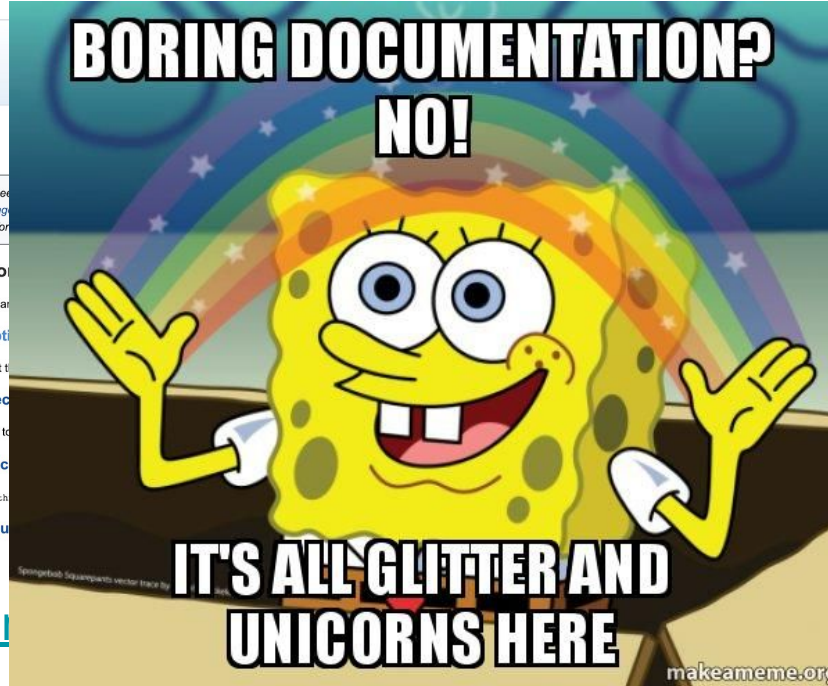
The Catch or Specify

This section covers how to

How to Throw Exc

This section covers the us

The try-with-resou



Hide TOC Search

Home Page > Essential Classes

nger available.

<https://docs.oracle.com>

[index.html](#)

Let's warm up

Is the following code legal?

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

Warm up exercises

What exception types can be caught by the following handler?

```
catch (Exception e) {  
    e.printStackTrace();  
}
```

What is wrong with using this type of exception handler?

Warm up exercises

Is there anything wrong with the following exception handler as written? Will this code compile?

```
try {  
    } catch (Exception e) {  
    } catch (ArithmeticException a) {  
    }
```


Warm up exercises

Match each situation in the first list with an item in the second list.

- A. `int[] A;`
`A[0] = 0;`
- B. The JVM starts running your program, but the JVM can't find the Java platform classes. (The Java platform classes reside in `classes.zip` or `rt.jar`.)
- C. A program is reading a stream and reaches the `end of stream` marker.
- D. Before closing the stream and after reaching the `end of stream` marker, a program tries to read the stream again.

- 1. `__error`
- 2. `__checked exception`
- 3. `__compile error`
- 4. `__no exception`

Exercise 1

(Extend your program from lab 5)

Write a program that reads the data from a text file and writes it into another text file. Handle following exceptions:

- input file does not exist
- no write-permission for output file

Exercise 2

Write a program that reads an input from passed arguments (NOT from the console user typed), which accepts two integer parameters, and divides the first integer by the second. You have to catch all the possible exceptions (such as, parsing errors, non-integer errors, arithmetic errors), and print the appropriate message.

Exercise 3

The method below downloads an image (actually any file). Your task is to edit the code so that it could handle all the possible exceptions.

```
public static void saveImage(String imageUrl) {
    URL url = new URL(imageUrl);
    String fileName = url.getFile();
    String destName = "./figures" + fileName.substring(fileName.lastIndexOf("/"));
    System.out.println(destName);

    InputStream is = url.openStream();
    OutputStream os = new FileOutputStream(destName);

    byte[] b = new byte[2048];
    int length;

    while ((length = is.read(b)) != -1) {
        os.write(b, 0, length);
    }

    is.close();
    os.close();
}
```

Exercise 4

Extend your “Equipment rental” system adding feature to *serialize* and *deserialize* the equipment list (use *java.io.Serializable* for that purpose). Handle all the exceptions that might occur in the process.

References

- https://www.tutorialspoint.com/java/java_exceptions.htm
- <https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/errors.html>
- <https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html>
- <https://www.baeldung.com/java-exceptions>
- https://www.protechtraining.com/content/java_fundamentals_tutorial-exceptions