



MODULE 1-05

Compact Course Programming. Java

- Organisation and Introduction -



Brief Introduction...

Prof. Dr. Galyna Tabunshchuk
Guest Professor
Computer Science Department
Fachhochschule Dortmund



Contact Details

- galyna.tabunshchuk@fh-dortmund.de
- Office: Otto-Hahn-Straße , Room 1.10

Course Target: Learn basic skills and concepts of computer programming in an object-oriented approach using Java. At the end of the course you will be able to write Java programs.

Course Prerequisites: Basic Java knowledge from pre-course is required for this course. Also, some basic knowledge and experience with computer systems is required.

Course Topics:

- Exception handling
- Basic input-output
- Concurrency
- Regular expressions
- Unit testing



-
- This course is based on highly interactive lectures
 - with a lot of content
 - with small quizzes, tasks and exercises for each student during the lecture
 - with student homework solution presentations
 - This course requires relevant self-study time (homework assignments, further reading etc.)
 - How to be successful in that course
 - Attend all lectures, concentrate on lecture content
 - Actively apply lecture content in lecture quizzes, tasks, exercises etc.
 - Review lecture content & exercises after the lecture & identify questions
 - Solve and submit the homework assignments
 - In the next lecture: clarify questions & present your homework assignment



Compact Course Exam - Presentation of the capstone project



Additional Grade Points

- Self-study work on homework assignment and the active participation in class are very important

Date	Type of the study	Type of Assignment
Friday, October 6, 2023	selfstudy	
Friday, October 13, 2023	selfstudy	
Friday, October 20, 2023	on-site	Introduction to the course.
Friday, October 27, 2023	selfstudy	
Friday, November 3, 2023	ILIAS upload deadline	Hometask1.
Friday, November 10, 2023	on-site	QA session
Friday, November 17, 2023	ILIAS upload deadline	Hometask 2.
Friday, November 24, 2023	selfstudy	
Friday, December 1, 2023	ILIAS upload deadline	Hometask 3.
Friday, December 1, 2023	onsite	QA session
Friday, December 8, 2023	selfstudy	Knowledge test 2. OOP
Friday, December 15, 2023	ILIAS upload deadline	Hometask 4.
Friday, December 22, 2023	onsite	QA session
Friday, December 29, 2023	ILIAS upload deadline	Hometask 5.
Friday, January 5, 2024	onsite	QA session
Friday, January 12, 2024	selfstudy	
Friday, January 19, 2024	selfstudy	
Friday, January 26, 2024	onsite	QA session
Friday, February 2, 2024	ILIAS upload deadline	Capstone project and All Hometask assignment
Friday, February 9, 2024	onsite	Final presentation

OER such as The Java™ Tutorials :

ORACLEJava Documentation

The Java™ Tutorials

« Previous • Trail • Next »

Home Page

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.

Trail: Essential Java Classes

This trail discusses classes from the Java platform that are essential to most programmers.

- Exceptions** explains the exception mechanism and how it is used to handle errors and other exceptional conditions. This lesson describes what an exception is, how to throw and catch exceptions, what to do with an exception once it has been caught, and how to use the exception class hierarchy.
- Basic I/O** covers the Java platform classes used for basic input and output. It focuses primarily on *I/O Streams*, a powerful concept that greatly simplifies I/O operations. The lesson also looks at *Serialization*, which lets a program write whole objects out to streams and read them back again. Then the lesson looks at some file system operations, including random access files. Finally, it touches briefly on the advanced features of the New I/O API.
- Concurrency** explains how to write applications that perform multiple tasks simultaneously. The Java platform is designed from the ground up to support concurrent programming, with basic concurrency support in the Java programming language and the Java class libraries. Since version 5.0, the Java platform has also included high-level concurrency APIs. This lesson introduces the platform's basic concurrency support and summarizes some of the high-level APIs in the `java.util.concurrent` packages.
- The Platform Environment** is defined by the underlying operating system, the Java virtual machine, the class libraries, and various configuration data supplied when the application is launched. This lesson describes some of the APIs an application uses to examine and configure its platform environment.
- Regular Expressions** are a way to describe a set of strings based on common characteristics shared by each string in the set. They can be used to search, edit, or manipulate text and data. Regular expressions vary in complexity, but once you understand the basics of how they're constructed, you'll be able to decipher (or create) any regular expression. This lesson teaches the regular expression syntax supported by the `java.util.regex` API, and presents several working examples to illustrate how the various objects interact.



MODULE 1-05: Compact Course Programming

Project requirements

eclipse

Eclipse IDE ▾ Working Group ▾ Projects Resources ▾ 🔍 👤 ▾

ECLIPSE IDE

The Leading Open Platform for Professional Developers

[Download 2023-09](#)

[Other Packages](#) [Sponsor](#)

Register for TheiaCon 2023!

Join us November 15-16 for our virtual conference on the Eclipse Theia IDE ecosystem, hosted by the Eclipse Cloud DevTools Working Group.

[Register Today!](#)

Eclipse Installer 2023-09 R

The Eclipse Installer 2023-09 R now includes a JRE for macOS, Windows and Linux.

Try the Eclipse Installer 2023-09 R

The easiest way to install and update your Eclipse Development Environment.

📄 1,382,210 Installer Downloads
📄 1,518,823 Package Downloads and Updates

Download

macOS [x86_64](#) | [AArch64](#)
Windows [x86_64](#)
Linux [x86_64](#) | [AArch64](#)

1

- Create solution

2

- Commit it to the GitHub repository

3

- Record screencast

4

- Answer questions

5

- Upload to the Ilias zip file named as teamHA#.zip which contains links to the screencast and solution

On-site presentation on 9 February

Evaluated:

- originality of the solution – 30%,
- individual impact of each team member 30%,
- knowledge in the object-oriented programming 30%,
- quality of the presentation 10%



MODULE 1-05: Compact Course Programming

Part 1. Exception Handling

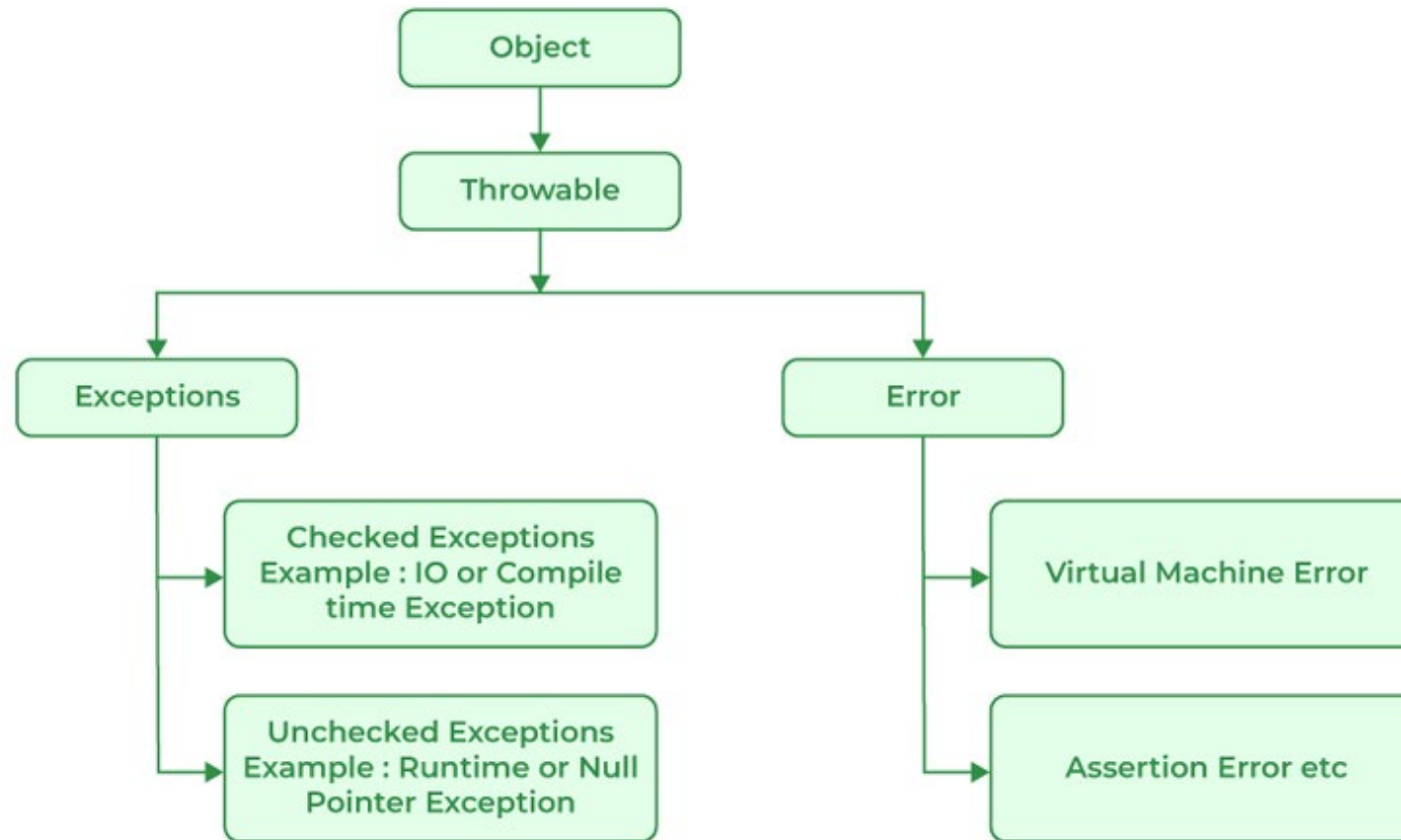
Java specific keywords for exception handling purposes

- **throw** – We know that if an error occurs, an exception object is getting created and then Java runtime starts processing to handle them. Sometimes we might want to generate exceptions explicitly in our code. For example, in a user authentication program, we should throw exceptions to clients if the password is null. The throw keyword is used to throw exceptions to the runtime to handle it.
- **throws** – When we are throwing an exception in a method and not handling it, then we have to use the throws keyword in the method signature to let the caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate them to its caller method using the throws keyword. We can provide multiple exceptions in the throws clause, and it can be used with the main() method also.
- **try-catch** – We use the try-catch block for exception handling in our code. try is the start of the block and catch is at the end of the try block to handle the exceptions. We can have multiple catch blocks with a try block. The try-catch block can be nested too. The catch block requires a parameter that should be of type Exception.
- **finally** – the finally block is optional and can be used only with a try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use the finally block. The finally block always gets executed, whether an exception occurred or not.

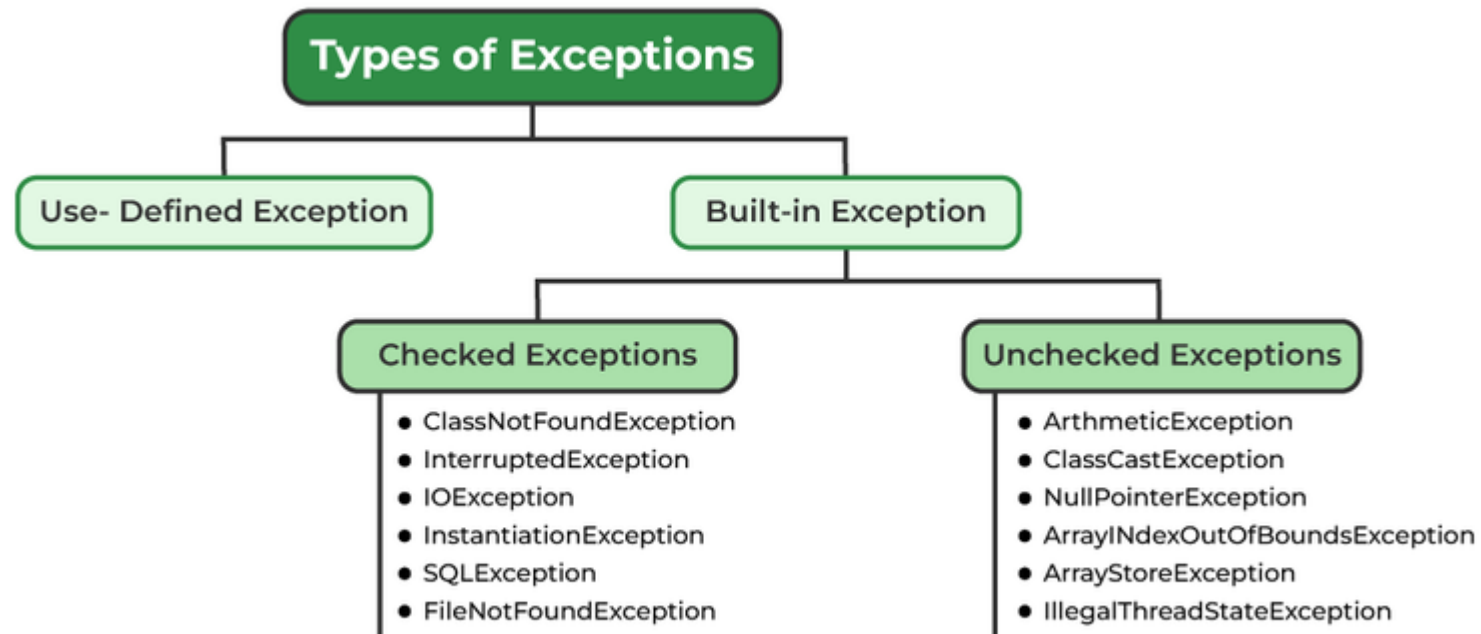
Reasons for Exceptions

- Invalid user input
 - Device failure
 - Loss of network connection
 - Physical limitations (out-of-disk memory)
 - Code errors
 - Opening an unavailable file
- **Error:** An Error indicates a serious problem that a reasonable application should not try to catch.
 - **Exception:** Exception indicates conditions that a reasonable application might try to catch.

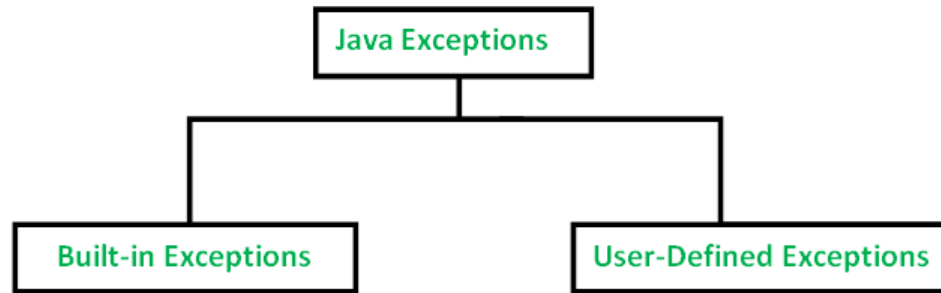
Exception Hierarchy



Types of exceptions



<https://media.geeksforgeeks.org/wp-content/uploads/20230714113547/Exceptions-in-Java-1-768.png>



```
class ArithmeticException_Demo
{
    public static void main(String args[])
    {
        try {
            int a = 30, b = 0;
            int c = a/b; // cannot divide by zero
            System.out.println ("Result = " + c);
        }
        catch(ArithmeticException e) {
            System.out.println ("Can't divide a number by 0");
        }
    }
}
```

<https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/>

- Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.
 1. **ArithmeticException:** It is thrown when an exceptional condition has occurred in an arithmetic operation.
 2. **ArrayIndexOutOfBoundsException:** It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
 3. **ClassNotFoundException:** This Exception is raised when we try to access a class whose definition is not found
 4. **FileNotFoundException:** This Exception is raised when a file is not accessible or does not open.
 5. **IOException:** It is thrown when an input-output operation failed or interrupted
 6. **InterruptedException:** It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.
 7. **NoSuchFieldException:** It is thrown when a class does not contain the field (or variable) specified
 8. **NoSuchMethodException:** It is thrown when accessing a method that is not found.
 9. **NullPointerException:** This exception is raised when referring to the members of a null object. Null represents nothing
 10. **NumberFormatException:** This exception is raised when a method could not convert a string into a numeric format.
 11. **RuntimeException:** This represents an exception that occurs during runtime.
 12. **StringIndexOutOfBoundsException:** It is thrown by String class methods to indicate that an index is either negative or greater than the size of the string
 13. **IllegalArgumentException :** This exception will throw the error or error statement when the method receives an argument which is not accurately fit to the given relation or condition. It comes under the unchecked exception.
 14. **IllegalStateException :** This exception will throw an error or error message when the method is not accessed for the particular operation in the application. It comes under the unchecked exception.

User-Defined Exceptions

```
public class MyException extends Exception {  
  
    private static final long serialVersionUID =  
        4664456874499611218L;  
  
    private String errorCode = "Unknown_Exception";  
  
    public MyException(String message, String  
        errorCode) {  
        super(message);  
        this.errorCode=errorCode;  
    }  
  
    public String getErrorCode() {  
        return this.errorCode;  
    }  
}
```

Home task1.

-
1. Read the tutorial
<https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html>
 2. Give answers on the following the tasks:
 1. Give the example of the **try-with-resources Statement**
 2. What is the purpose of throw statement
 3. Which classes inherit from Throwable class
 4. Give 5 different examples of catching and handling exceptions
 1. Create the class for exception handling in your project. Distribute the task in your team so that each team member responsibility to cover the topic:
 - a. **Handling Multiple Exceptions**
 - b. **Re-throwing Exceptions:**
 - c. **Resource Management:**
 - d. **Chaining Exceptions**
 2. Upload your project to the GitHub.
 3. Record 8 min screencast of your work. Upload to the personal YouTube channel
 4. Submit a link to the GitHub project and a link to the YouTube video.