# Classes, Objects, Constructors and Encapsulation

Louis Botha, louis.botha@tuni.fi

## github-classClassroom Assignment

* [Acceptance Link](https://classroom.github.com/a/cKkFaTvV)

## Gradle

The structure of the exercises are now following the java packaging standard.

❯ tree

.

├── README.md

└── e01

    ├── build.gradle

    └── src

        ├── main

        │   └── java

        │       └── com

        │           └── example

        │               └── project

        │                   ├── Book.java

        │                   └── Main.java

        └── test

            └── java

                └── com

                    └── example

                        └── project

                            └── BookTest.java

Exercises can be build and tested with gradle commands.

To build the project:

❯ gradle build

> Task :test

BookTest > testPages() PASSED

BookTest > testTitle() PASSED

BookTest > testReadingTime() PASSED

BookTest > testAuthor() PASSED

BUILD SUCCESSFUL in 1s

4 actionable tasks: 4 executed

To execute the tests:

❯ gradle test

> Task :test

BookTest > testPages() PASSED

BookTest > testTitle() PASSED

BookTest > testReadingTime() PASSED

BookTest > testAuthor() PASSED

BUILD SUCCESSFUL in 1s

3 actionable tasks: 3 executed

Running the application:

// Change to the build directory

❯ cd build/classes/java/main

❯ java Main

**OR**

You can still build and run the program without gradle. You need to cd into the src\main\java\com\example directory and use javac and java to compile and run the applications.

## laptop computer01

1. Create a class called "Book" that has properties for title, author, and number of pages.
2. Add methods to set and get the values of the properties, as well as a method to calculate the total reading time based on an average reading speed of 200 words per minute.

public double calculateReadingTime() {

        // Assuming an average reading speed of 200 words per minute

        int avgWordsPerPage = 250;

        System.out.println((pages \* avgWordsPerPage) / 200.0);

        return (pages \* avgWordsPerPage) / 200.0;

}

1. Create multiple objects of the Book class and test the methods.

## laptop computer02

1. Create a new Java class called “Car" that has the following properties:
   * make (String)
   * model (String)
   * year (int)
   * speed (int)
   * isRunning (boolean)
2. Use the private access modifier for all the properties.
3. Create getter and setter methods for each property.
4. Create a method called "startEngine" that sets the isRunning property to true.
5. Create a method called "stopEngine" that sets the isRunning property to false.
6. Create a method called "accelerate" that increases the speed of the car by 10.
7. Create a method called "brake" that decreases the speed of the car by 10.
8. In the main method, create an object of the Car class and set the properties using the setter methods.
9. Use the getter methods to print the properties of the car object.
10. Use the startEngine, accelerate and brake methods to change the state of the car.
11. Use the getter methods to print the properties of the car object after the changes.
12. Add a validation to the setter methods to make sure that the properties are not set to an invalid value.
13. Add a validation to the accelerate and brake methods to make sure that the speed does not go above or below a certain limit.
14. Add a validation to the startEngine method to make sure that the car is not already running.
15. Add a validation to the brake method to make sure that the speed is not already zero.

## laptop computer03

1. Create a class called "Person" that has private properties for
   1. first name,
   2. last name,
   3. and age.
2. Add getters for each of the properties.
3. Add a constructor to the class that takes in the first name, last name, and age as parameters and sets the corresponding properties.
4. Add a second constructor that takes in only the first name and last name as parameters and sets the corresponding properties and age to 0.
5. Add a third constructor that takes no parameters and sets all properties to their default values.
6. Create multiple objects of the Person class using all the constructors and print their properties.
7. Add a method to the class called "display" that prints the properties of the object.
8. Create a main method to test all the constructors and the display method.

## laptop computer04

1. Create a class called "Animal" that has properties for name, type and age.
2. Override the toString() method in the class so that it returns a string in the format

* "Name: [name], Type: [type], Age: [age]".

1. Override the equals() method in the class so that it compares the name, type and age of the objects.
2. Create multiple objects of the Animal class and test the toString() and equals() methods by comparing them and printing them to the console.
3. Add a method to the class called "display" that prints the properties of the object using the toString method.
4. Create a main method to test the toString, equals and display methods.