**Programare Avansata pe Obiecte  
Laborator 4**

**Pahontu Bogdan-Ionut**

**E-mail:** [**pahontubogdan@gmail.com**](mailto:pahontubogdan@gmail.com)

Contents

[1. Immutable classes 3](#_Toc3847929)

[1.1. What is an immutable class 3](#_Toc3847930)

[1.2. How to create an immutable class 3](#_Toc3847931)

[1.3. Properties of immutable classes 3](#_Toc3847932)

[2. String class 4](#_Toc3847933)

[3. StringBuilder and StringBuffer classes 4](#_Toc3847934)

[4. Regular expressions 5](#_Toc3847935)

[4.1. General aspects 5](#_Toc3847936)

[5. Abstract classes 6](#_Toc3847937)

[5.1. Properties 6](#_Toc3847938)

[6. Tasks 6](#_Toc3847939)

# Immutable classes

## What is an immutable class

* Immutable class means that once an object is created, we cannot change its content;
* Immutable classes are helpful because you know they will always be the same;
* You can pass them around your application with a guarantee that the caller didn’t change anything;
* All wrapper classes are immutable;
* We still want the caller to be able to specify the initial value—we just don’t want it to change after the object is created;

## How to create an immutable class

* Class must be declared as final (So that child classes can’t be created);
* Data members in the class must be declared as final (So that we can’t change the value of it after object creation);
* A parameterized constructor;
* If in the constructor we receive a mutable variable, we need to create a copy of it;
* Getter method for all the variables in it;
* No setters (To not have option to change the value of the instance variable);

## Properties of immutable classes

* Simple to construct, test and use
* Thread safe;

**public final class** ImmutableExample {  
 **final** String **name**;  
 **final int regNo**;  
  
 **public** ImmutableExample(String name, **int** regNo) {  
 **this**.**name** = name;  
 **this**.**regNo** = regNo;  
 }  
  
 **public** String getName()  
 {  
 **return name**;  
 }  
 **public int** getRegNo()  
 {  
 **return regNo**;  
 }  
}

# String class

* String class is another good example of immutable class and it is more like a utility class.
* Any action that will be perform on an object of String type will conduct to another String object.
* String Methods:
  + length() – return the number of the characters of the string;
  + concat() – concatenating two strings;
  + format() – formatting a string;
  + split() - Splits this string around matches of the given regular expression;
  + indexOf() - Returns the index within this string of the first occurrence of the specified character/string;
  + lastIndexOf() - Returns the index within this string of the last occurrence of the specified character/string;
  + toLowerCase() - Converts all of the characters in this String to lower case using the rules of the default locale;
  + toUpperCase() - Converts all of the characters in this String to upper case using the rules of the default locale;
  + substring() - Returns a new string that is a substring of this string;
  + startsWith() - Tests if this string starts with the specified prefix;
  + endsWith() - Tests if this string ends with the specified prefix;

More String functions can be found at: <https://docs.oracle.com/javase/8/docs/api/index.html> - search by String class;

# StringBuilder and StringBuffer classes

* StringBuilder in Java represents a mutable sequence of characters. Since the String Class in Java creates and immutable sequence of characters, the StringBuilder class provides an alternate to String Class, as it creates a mutable sequence of characters
* StringBuffer is similar to StringBuilder class;
* However the StringBuilder class differs from the StringBuffer class on the basis of synchronization;
* The most usual method used for StringBuilder is **append**, that will allow user to add a string at the end of other string;
* Other StringBuilder methods:
  + insert() – insert the given string in another string at the given position;
  + replace() - method replaces the given string from the specified beginIndex and endIndex;
  + delete() - deletes the string from the specified beginIndex to endIndex;
  + reverse() - reverses the current string;

Other information at:

* <https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html>
* <https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuffer.html>



Figure 1 – String vs StringBuilder vs StringBuffer

* Java String Pool info: <https://www.journaldev.com/797/what-is-java-string-pool>

# Regular expressions

## General aspects

* A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. They can be used to search, edit, or manipulate text and data.
* You can test your regular expressions on: [https://regex101.com](https://regex101.com/)
* More examples:
  + <https://docs.oracle.com/javase/tutorial/essential/regex/>
  + <https://www.tutorialspoint.com/java/java_regular_expressions.htm>
* Java provides the java.util.regex package for pattern matching with regular expressions.
  + Pattern Class − A Pattern object is a compiled representation of a regular expression;
  + Matcher Class − A Matcher object is the engine that interprets the pattern and performs match operations against an input string.

# Abstract classes

## Properties

* An abstract class is declared using abstract keyword;
* It can have bot abstract and non-abstract methods;
* Abstract classes can`t be instantiated;
* Can have constructor;
* Can have static methods;
* Can have final methods;
* For each abstract method we will have a concrete implementation in the concrete class;

# Tasks

**Task 1:**

1. Create packages lab4 -> tasks
2. Create a package lab4 🡪 tasks🡪task1;
3. Create a main class in which will be used the following string methods:
   1. indexOf, lastIndexOf, length, split;

**Task 2:**

1. Write a function that will remove any leading whitespace from a given string- hint: trim;

**Task 3:**

1. Write a function that will concatenate 2 given strings if the first one starts with the last three letters from the second one(from end to start);
   1. Ex: String1: “Ele fac” , string2: “cafele”;

**TASK 4:**

1. In the project that was started at lab 3 add two new classes Invoice și InvoiceItem in the models package;
2. In the invoice entity add fields like invoiceDate, invoiceName and an array of invoice items;
3. The invoiceItem entity will have a product field and a TVA field marked as double;

**TASK 5:**

1. Change the class Product and make it abstract, add two, three abstract methods (ex: roundPrice, etc..)
2. Create classes that extend Product class:
   1. FoodProduct;
   2. ClotheProduct;
   3. CleaningProduct;
   4. FurnitureProduct;
3. For each of this classes rewrite method toString using StringBuilder;

**TASK 6:**

1. Create a new package named configs;
2. In this package create a new immutable class named Configuration. This class will have fields like: dbConnectionString, externalResourcesPath, environment;
3. Implement the required methods;
4. Create two new Configuration instances one for production and one for development.

**TASK 7:**

1. Write a function to find the second most frequent character in a given string.
   1. The given string is: successes
   2. The second most frequent char in the string is: c

**TASK 8:**

1. Look over the documentation mentioned above about String, StringBuilder and String Buffer;
2. Use 10 of the class functions in some basics examples.