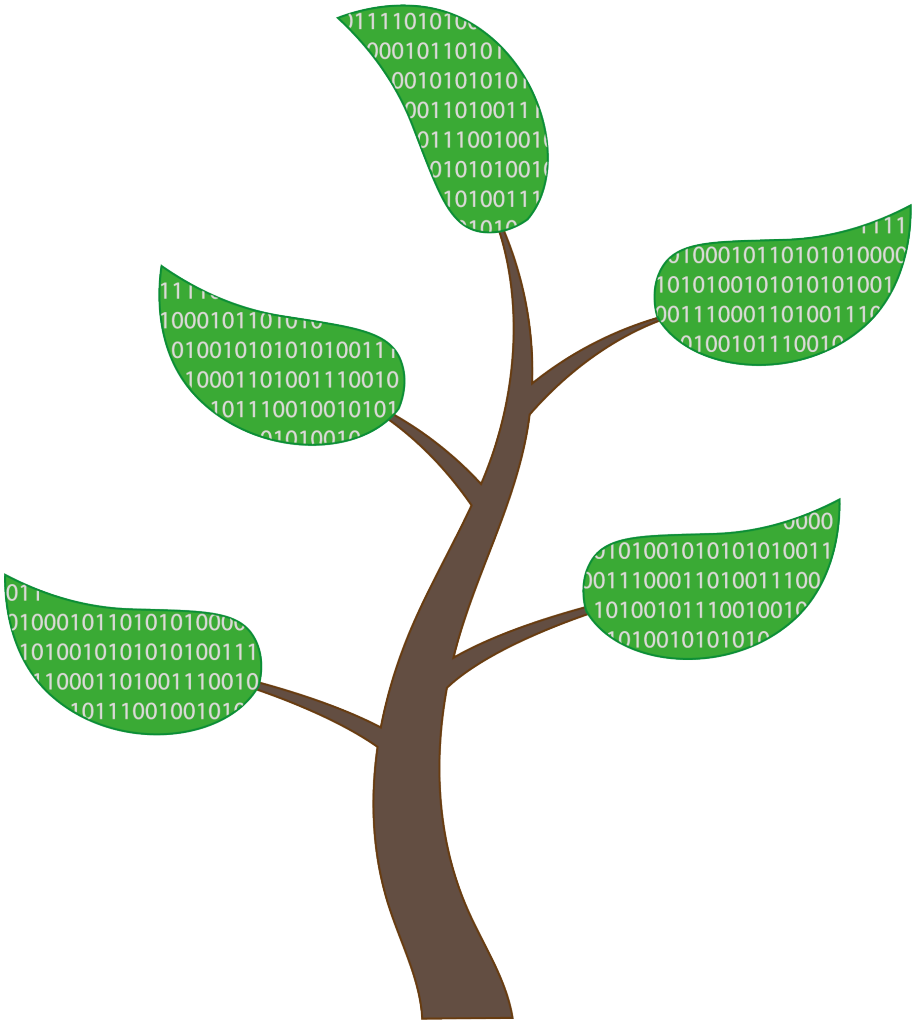

# Programming Exercises for NoBeginners*

PART **1**

By Alexander Tyshchenko
http://tyshchenko.pro

February 1, 2022

# Contents

# About author

**Alexander Tyshchenko** is a consulting engineer with over 10 years of experience in software development. He also **Oracle Java Certified Programmer** and **Individual Member** at jcp.org (Java Community Process).

# Introduction

## Who is this book for

## What is covered in this book

## What you need to use this book

## Conventions

# Chapter 1

# Beginning with OOP

**1.** Create class Father (with no properties and methods), extends class Father with class Son.

**2.** Create class Person and define fields:

- first name
- second name
- date of birth
- current age
- living address
- sex
- profession
- is married
- amount of children

Also create methods for settings and getting for all fields.

**3.** Create class MobilePhone and define fields:

- model
- memory
- weight
- amount of sim slots
- screen size
- screen resolution

- color
- back camera pixels
- front camera pixels
- max photo resolution
- max video resolution
- processor frequency
- processor cores
- wi-fi standard version
- bluetooth standard version
- is NFC presents
- is GPS presents
- power type connector

Also create functions for settings and getting for all fields.

**4.** Create class Circle2D, define fields (radius, x - coordinate, y - coordinate), create functions for settings and getting fields

**5.** Create class House, define following below fields:

- square of the house
- number of living rooms
- number of bathrooms
- number of persons who live in the house
- number of floors
- is presents elevator in the house
- address of the house
- country where the house is located

Also create functions for settings and getting for all fields.

**6.** Create class Tree with fields:

- amount of leafs
- number of branches
- tree species

- current age
- height
- tree trunk diameter

Also create functions for settings and getting for all fields.

**7.** Create class Car with fields:

- model
- color
- number of seats
- volume of the engine
- is manual transmission
- amount of speeds
- volume of the tank

Also create functions for settings and getting for all fields. Create methods with empty code implementation:

- open the car door
- close the cat door
- fasten the seat belt
- turn on the engine
- press the gas pedal

**8.** Create class Television with methods:

- turn on tv
- turn off tv
- increase volume
- decrease volume
- change channel up
- change channel down

**9.** Create class CoffeeMachine and define all possible fields and methods

**10.** Create class ComputerMouse and define fields and methods:

- numbers of buttons
- numbers of scrolls
- DPI value
- click button
- clock scroll
- model
- weight
- size
- color
- year of assembling
- serial number
- connection type

**11.** Create class ComputerKeyboard and define fields and methods:

- numbers of keys
- color
- is mechanical
- is fully sized
- year of assembling
- serial number
- connection type

**12.** Create class ComputerSystemBody and define fields and methods:

- color
- size
- numbers of processors
- motherboard models
- motherboard processor socket
- processor frequency
- memory size
- disk memory
- graphics card model

- list of supported monitor sockets
- numbers of USB sockets

**13.** Create class **Robot**. Add methods:

- move forward
- move back
- move left
- move right

When calling methods also need to print to the console action message. Create object of class **Robot** and call methods:

- 4 times move left
- 1 time move back
- 10 times move right
- 2 times move forward

**14.** Create classes **Fruit**, **Apple**, **Pear**, **Plum**. All fruit classes must extend class **Fruit**. Create **3** objects:

- first from class Apple
- second from class Pear
- third from class Plum

All objects must be placed into one array.

**15.** Create classes **Table** and **TableLeg**. **Table** class object must contain **3** or more **TableLeg** objects.

**16.** Create class **Airplane** and **methods**:

- take-off
- flight
- landing

Implement all methods that just output to the console. Create an object of class **Airplane** and call all three methods.

**17.** Create class **Flower** and class **Petal**. A **Flower** object can contain multiple **Petal** objects. **Petal** class must have properties:

- color
- shape
- thickness

Create an object of **Flowers** class with 8 petals;

**18.** Create classes **Planet**, **SolarSystem** and **Star**. **SolarSystem** must contains **planet objects** and **star** (SunStart). Planet class must contain fields:

- name
- volume
- weight
- distance from the star

**19.** Create classes **Wall**, **Window** and **Door**. A Wall can have multiple windows and doors. Create a **wall object** that contains **3** windows and **2** doors.

**20.** Use classes from previous exercise. Create a class **Room**. A Room can have **4** or more **walls** objects. Create a **room object** that contains **4** walls:

- First and Second wall must contain **0** windows and doors
- Third wall must contain **3** windows and **1** door
- Fourth wall must contain **1** window and **1** door

**21.** Write a program that fill bookshelf by books.

- Create class **BookShelf** with fields (books, shelfWidth, freeSpaceWidth, busySpaceWidth)
- Create class **Book** with fields (width, pages, name)

You goal is to fill bookshelf object by books objects. Every book must have different width. Program must check if it possible to add new book object to a bookshelf.

**22.** Create a program that post job resumes on a free web portal. A class **WebPortal** will represents all posted resumes with one method **postA-Job** that accept object of a class **Resume**. Class **Resume** must contain following below fields:

- full name
- sex
- year of birth
- years of experience
- skills
- wished salary

All fields in class **Resume** are mandatory. **WebPortal** object must do not accept **Resume** object with empty fields.

**23.** Create classes **Person** (with field **name**) and **Apple**. Then create 2 persons objects (instances of **Person**) with different names and 1 object of **Apple**. The main idea is to exchange object of an **Apple** between to objects of **Person**. First add apple object into first person. Than first person must hand over an apple to the second person. And vice versa.

**24.** Create class **Refrigerator** that can contain multiple products. Create class **Product** with following below fields:

- name
- expiration date
- weight

Create classes **Cheese**, **Sausage**, **Fish**, **Meat**, **Egg** by extending from a class **Product**. Create different objects for all product classes and add them to **Refrigerator** object.

**25.** Write a program that can operate with 3-dimensions vectors

- create a class Vector3D
- create filed for 'x' coordinate
- create filed for 'y' coordinate
- create filed for 'z' coordinate
- implement vector's operations (+, -, length)

**26.** Write a program that can operate with fraction numbers

- create a class Fraction
  - create a field numerator

  - create a filed denominator
  - create constructor Fraction(numerator, denominator)
  - implement all arithmetic operations (+, -, *, /)

**27.** Write a program that can operate with complex numbers

- create a class ComplexNumber
  - create a field realPart
  - create a filed imaginaryPart
  - create constructor ComplexNumber(realPart, imaginaryPart)
  - implement all arithmetic operations (+, -, *, /)

**28.** Write a program that create hierarchy of geometric figures. For different classes will be different constructors.

- create abstract class Figure with methods
  - getArea
  - getPerimeter
  - getName
- create class Rectangle (extend class Figure) and override all methods from class Figure
- create class Circle (extend class Figure) and override all methods from class Figure
- create class Rhombus (extend class Figure) and override all methods from class Figure
- create class Triangle (extend class Figure) and override all methods from class Figure
- create class Square (extend class Figure) and override all methods from class Figure
- create class Trapeze (extend class Figure) and override all methods from class Figure

**29.** Write a program that create students database

- create a class Person with fields
  - first name
  - last name
  - sex

  - date of birth
- extends class Person with class Student, add additional fields
  - speciality
  - department
  - average rate
- create class StudentDatabase and following below method implementation
  - findAll
  - findByFirstName
  - findByLastName
  - findBySex
  - findBySpeciality
  - findByDepartment
  - findByAverageRate

**30.** Write a program that create library (books) database

- create a class Book with fields
  - title
  - pages
  - author
  - category
  - isbn
- create a class LibraryDatabase and following below method implementation
  - findAll
  - findByTitle
  - findByPages
  - findByAuthor
  - findByCategory
  - findByISBN

# Chapter 2

# Multithreading

**1.** Write a program that calculate in separated thread $\sqrt[2]{100}$ and print to the console.

**2.** Write a program that print numbers from **1** to **10** (use any loop) in separated thread.

**3.** Write a program that create **2** threads. The first thread must generate random number from 10 to 30 and print to the console. The second thread must generate random number from 50 to 100 and print to the console. All thread must be run in the same time. Only need **3** numbers from each thread, sleep time between generation is **3** seconds.

**4.** Write a program that run **15** threads. Every thread must print to the console random value from **10** to **100**

**5.** Write a program that create **2** threads. Every thread after starting must wait **10** seconds and print to the console message **waiting finished** than thread must be stopped.

**6.** Write a program that create and run one thread that wait **4** seconds. After that create in the first thread second thread that must print to that console "Hello from the second thread".

**7.** Write a program that create one thread and after waiting **3** seconds print to the console the name of the thread. After that thread must create and run other thread with the same login. Repeat such logic **5** times.

**8.** Write a program that create **10** threads that must print on a console the message - "Hello from a thread (name=?)", where **?** in the message must be thread name. Run all **10** threads.

**9.** Write a program that check when an array will be filled fully:

- create an empty array with 10 elements.
- create two threads (one for filling values and one for checking if arrays is filled fully).
- run all two threads.
- first thread must put new random value into array and sleep for **2** seconds.
- second thread must check if all array is filled than print all values, if not than sleep for **3** seconds and check again

**10.** Write a program that make operation with a bank account. Create a bank account class, place default money amount (1000 US dollars). Than make 10 sequent deposit operations and 5 withdrawal sequent operations. All operations must be run in separated threads.

# Chapter 3

# Network

1.

# Chapter 4

# Algorithms

1.

# Chapter 5

# Relational Databases

**1.** Write a sql query that create a table

| ID | FIRS_NAME | SECOND_NAME | EMAIL |
|---|---|---|---|

# Chapter 6

# Web Services

1.

# Chapter 7

# Projects for Portfolio

1. Write a web application that implement simple testing system with choices for any domain
    - relational database must contain all data (use any ORM framework for data access)
    - a test must contain multiple or one correct choices
    - no need any authorization or authentication
    - after and of a test application must show pass result
    - in test need to have forward and backward tests navigation
    - a test must have a time restriction
2. Write a client-server application - Trading system