Charles Sturt University

Assessment item 2 –

**Designing Object-Oriented Program**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Student Name: Alina Maksumova

Student ID: 11706271

Lecturer Name: Dr Robin Gao

Subject Name: Programming in Java 1

Subject Code: ITC538

Sydney 2020

**Contents**

[**Task 1** 3](#_Toc41510249)

[**Code description** 3](#_Toc41510250)

[**Output Screenshot** 4](#_Toc41510251)

[**Task 2** 6](#_Toc41510252)

[**Code description** 6](#_Toc41510253)

[**Output Screenshot** 6](#_Toc41510254)

[**Task 3** 9](#_Toc41510255)

[**Code description** 9](#_Toc41510256)

[**Output Screenshot** 9](#_Toc41510257)

[**Task 4** 10](#_Toc41510258)

[**Differences between paradigms** 10](#_Toc41510259)

[**Programming languages for paradigms** 11](#_Toc41510260)

[**Advantages of paradigm approaches** 11](#_Toc41510261)

[**Reference** 12](#_Toc41510262)

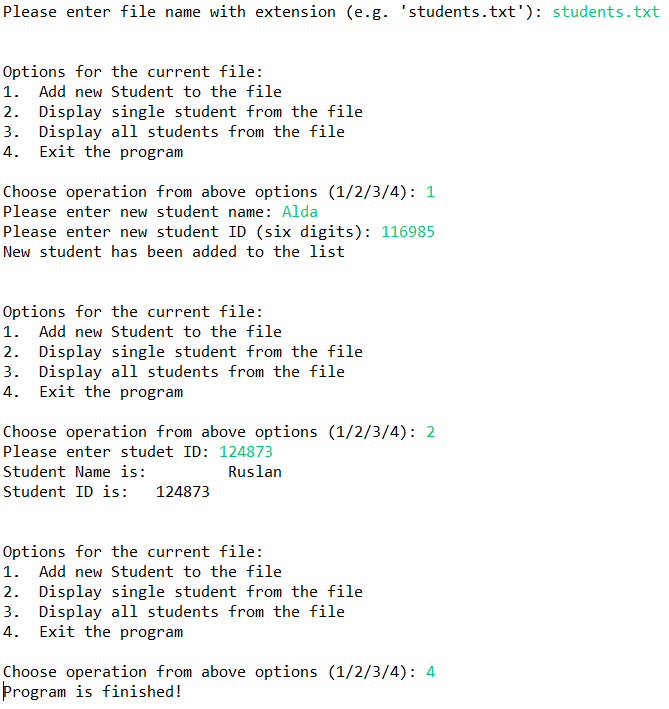
# **Task 1**

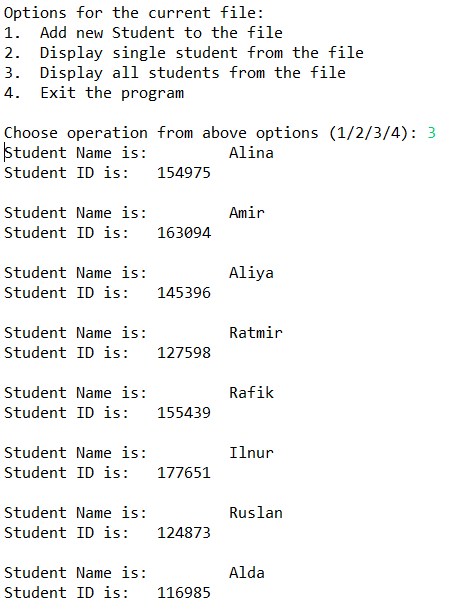
## **Code description**

There was created Student class, with student ID and name fields. There were also created get and set methods for these fields in order to set specific name or id and return them. In addition, there was created toString() method, which returned description representation of the object. Within the same class, there is method isValidId() to check if entered id is 6 digits long. Finally, IdExist() method checks if entered ID number is already in the file.

TestStudent class prompts user to enter file name to be opened, this file should be within the same directory as program itself, otherwise program cannot open it (program will display error message with the help of try/catch statement). The array list is created, composed from every next line of the file (lines are read by Scaner reader). The program will display menu of options to perform with the file: add new student, display single or all students, terminate the program. After user enters proper number for option 1/2/3/4), program will perform required task with the help of following methods: addNewStudent(), dislaySingleStudent (), displayAllStudents (), writeNewStudentToFile ().As a result, if student adds a new student, program will rewrite current file.

## **Output Screenshot**





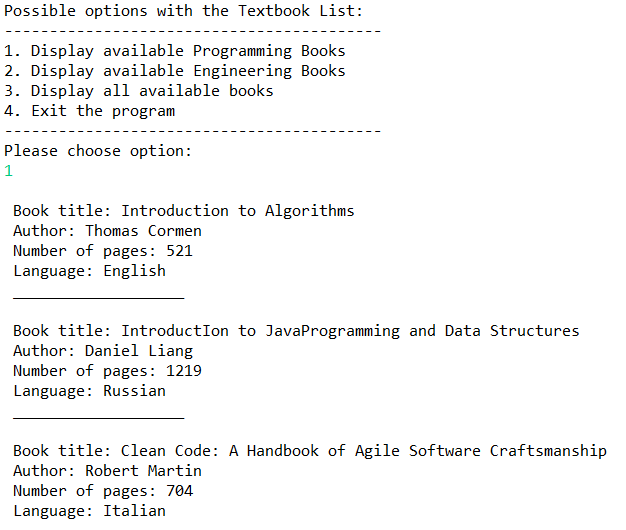
# **Task 2**

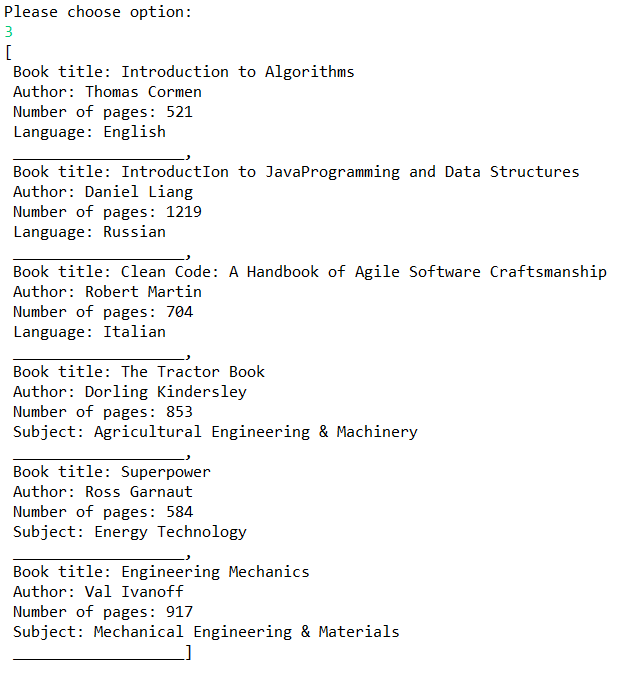
## **Code description**

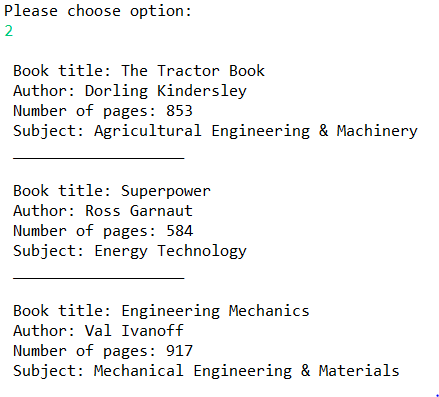
There was created abstract Textbook class with title, author and pages data fields. At the start, there is default constructor created. Besides, for each field there was created get and set methods, as well as toString() method, that displays object description to a user. Two subclasses, namely ProgrammingTextbook and EngineeringTextook were created. For the first subclass, there was created additional data field of textbook language, whereas for Engineering books there is additional field subject. Each of additional fields also have get and set methods to return or set variables respectively.

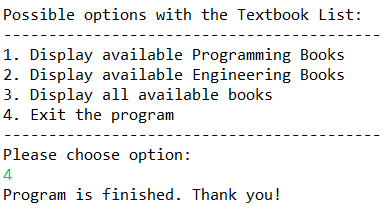
There was created TestTextbook to test the program. In this class, there was created an array list, which contains all textbooks. Moreover, there were created three new Programming, as well as three new Engineering Textbooks and added to the list. For each textbook, there were set title, author and number of pages as fields of superclass, as well as additional own field (language/subject) of a subclass. There was created a loop for user to choose operation: display engineering, programming or all textbooks, as well as option to terminate the program.

## **Output Screenshot**







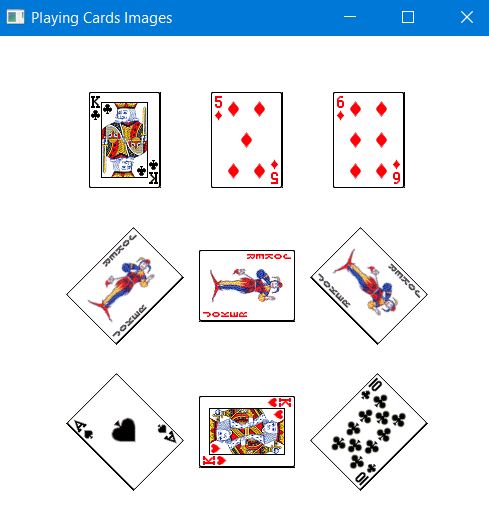


# **Task 3**

## **Code description**

For this task there was imported java Application class and other necessary utilities. There was created class Cards, which extends Application. In the beginning, there was declared an array list, which stores all the images to be displayed. To reference every card to the list, there was used for loop, where the source folder was declared (the whole path “file: ….”). To randomly choose cards, there was created random generator, which generates an integer, and this integer then serves as an index for the images from array list. In addition, Hbox and Vbox were used to display cards in the horizontal and vertical follows. Precisely, Hbox got 3 rows: in the first row 3 randomly selected images, in the second row 3 joker cards, in the third row 3 different cards. Moreover, for second and third rows setRotate() method was used to rotate cards to angle of 45/90/135 degrees, while for all rows setAllign() method was implied to align images to center. At the end, all three rows were vertically added with the help of Vbox. There was also imported Scene class and used to set borders of animation, namely 500x500 pixels width and height.

## **Output Screenshot**



# **Task 4**

## **Differences between paradigms**

Overall, paradigm in programming is a term which defines the way of how a certain issue can be resolved, involving specific tolls and practices (Simmonds, 2012). In general, paradigms can be divided into two large groups: Imperative and Declarative ones (Bempong, 2013). In order to discuss these groups, their constituents and difference between them, there is a following explanative table:

*Figure 1: Difference between programming paradigms*

|  |  |
| --- | --- |
| **Name** | **Description** |
| **Imperative Programming Paradigms**  Imperative paradigms present primary and old paradigms in the programming history. This type of paradigms changes state of the program and are usually simple in its implementation (Yolles & Fink, 2012). It is differentiated by its instructions provision, with step by step approach and detailed explanation. | |
| ***Procedural Programming Paradigm***  *(appeared in late 1950s)* | In the core of procedural paradigms, there lies a sequence of computational phases to be implemented. It is distinguished by set of subroutines that are about to be implied in a specific order. These paradigms support reuse of codes, which makes them useful for programming (Yolles & Fink, 2012). |
| ***Object Oriented Programming***  *(appeared in early 1960s)* | This paradigm is widely spread and is used by many programming languages. It does not present traditional way of programming, as it rather focuses on objects and classes (Narbel, 2009). Hereby, objects are presented by any nouns that have characteristics and behavior, while classes present blueprint of objects. This paradigm supports inheritance, polymorphism, abstraction, data security and many more features, including code reuse. |
| **Declarative Programming Paradigms**  In contrast to imperative programing, these type of paradigms do not declare the way of implementation. Therefore, it can accept various ways of problems solution (Lorenz, D., & Rosenan). Declarative paradigms are mainly focused on the result itself, not the way of achievement; and rely on logic approach to issues. | |
| ***Logic Programming Paradigm***  *(appeared in 1950s)* | This paradigm type is also known as abstract model and is focused on the logic of the program. As a base of coding implementation, this paradigm implies logical approach to tasks, while logic is based on programming knowledge (Simmonds, 2012). Consequently, logic of issues forms the programs and fed them to computer. |
| ***Functional Programming***  *(appeared in 1950s)* | This paradigm relies on mathematical approach to perform programming tasks, as well as it uses trees and expressions (Lorenz, D., & Rosenan, 2017). As a result, there are values returned from the functional procedures. Functional programming sometimes is called as modular approach programming, since codes are written in a composable manner (Bempong, 2013). In addition, this paradigm can be implied to a non-functional languages, such as JavaScript, PHP and several others. |

## **Programming languages for paradigms**

|  |  |
| --- | --- |
| **Paradigm Name** | **Programming Languages** |
| *Object Oriented Programming* | Simula, Java, Python, C++, C#, Ruby, JavaScript, Perl  While Simula was the first language to imply object oriented programming, other languages listed above are multi-paradigm. Therefore, major languages support OOP in line with other paradigms. These languages support objects, classes, encapsulation and lots of other important useful features (Narbel, 2009). |
| *Procedural Programming* | Pascal, C, Fortran, ALGOL, COBOL  Fortran, ALGOL, COBOL (1950s) appeared about 20 years earlier then Pascal and C(1970s). However, to call procedures, all these languages rely on stack register, i.e. hardware support (Lorenz, D., & Rosenan, 2017). |
| *Logic Programming* | Prolog, answer set programming (ASP), Datalog  Most common representative of logic programming is Prolog, which was developed in 1972. It relies on logic to understand semantics of the program (Yolles & Fink, 2012). Other languages are not that popular, however, follow same idea and non-deterministically provide issues solutions. |
| *Functional Programming* | LISP, IPL, APL, NPL  (non-functional PHP, Lua, JavaScript, C++, Kotlin)  Purely functional programming, which is presented by LISP, IPL, APL, NPL uses lambda notation (Bempong, 2013). Other listed languages are multi-paradigm. |

## **Advantages of paradigm approaches**

To discuss the benefits of paradigm approaches, they can be considered by previously mentioned two large groups: imperative and declarative paradigms.

Imperative programming:

1. It provides state of program execution. Therefore, it allows access to a set of program execution techniques (Simmonds, 2012). Paradigms of this type consider the way of problems solving, with detailed instructions (step by step implementation).
2. Another valuable benefit of imperative programming is simple legibility. Therefore, code created by this parading type can be easily understood by many users, who did not necessarily write it or contributed to it (Yolles & Fink, 2012).

Declarative Programming:

1. This type programs are known to provide compressed form of implementation, which makes the whole process of programming much quicker. This aspect also simplifies the process of writing complex codes (Simmonds, 2012).
2. Another important advantage of declarative programming is that it allows separation of development and maintenance of the program. Since programs rely on algorithms, new methods can be easily added to the system, as well as modified (Bempong,2013).

# **Reference**

Bempong, B. (2013). The Cognitive Programming Paradigm the Next Programming Structure. *American journal of software engineering and applications, 2*(2), 54. doi: 10.11648/j.ajsea.20130202.15

Lorenz, D., & Rosenan, B. (2017). Application Embedding: A Language Approach to Declarative Web Programming. *The art, science, and engineering of programming, 1*(1). doi: 10.22152/programming-journal.org/2017/1/2

Narbel, P. (2009). Functional Programming at Work in Object- Oriented Programming. *The journal of object technology, 8*(6), 181. doi: 10.5381/jot.2009.8.6.a5

Simmonds, D. (2012). The Programming Paradigm Evolution. *Computer, 45*(6), 93-95. doi: 10.1109/mc.2012.219

Yolles, M., & Fink, G. (2012). Understanding the Imperative for a Corporate Paradigm Change. *SSRN Electronic Journal.* doi: 10.2139/ssrn.1980790