

Linnaeus University

1DV532 – Starting out with Java

Assignment 2 Report

Student: Evan Huynh

Student ID: eh223im@student.lnu.se



Table of Contents

[Report 3](#_Toc44362345)

[Exercise 1: Int 3](#_Toc44362346)

[Idea 3](#_Toc44362347)

[Exercise 2: SweID 3](#_Toc44362348)

[Exercise 3: Pizza 3](#_Toc44362349)

[Exercise 4: Money 3](#_Toc44362350)

[Source code 3](#_Toc44362351)

[Bibliography 3](#_Toc44362352)

# Report

## Exercise 1: Int

The exercise requires basic arithmetic for two integers in “Int” type. For the plus and div methods, we simply add or divide each of their respected field and return type “Int”.

## Exercise 2: SweID

This exercise contains lots of static classes for Swedish ID and checkers. However, we assume that users will supply ID the form of YYMMDD-XXXX, otherwise we will passthrough *inputHandlingAndConvertingIntoTheCorrectFormToFurtherProcessingIDCorrectly* method and convert it into the correct form. This was based on the previous source code with improvement. [1] Since data is shared between every other method, all of them are static in the SweID class.

* *getFirstPart* and *getSecondPart*: implying that user supply the correct form of the ID (i.e. 010203-0405), simply getting first 6 characters and last 4 characters of the ID, respectively.
* *isFemaleNumber* checks the third digit of 4 last digits is divisible by 2, otherwise it is a male number.
* *areEqual* is *toString* but worse in performance.
* *isCorrect* checks if the ID is valid and requires 3 other methods: *isValidMonth, isValidDate, isValidChecksum*
  + *isValidMonth* checks if the month is between 01 to 12 (Gregorian calendar)
  + *isValidDate* check if the date is within the range of the month, meaning that if the month does not exist, *isValidDate* always returns false. It is also check if the year is leap to decide whether that February has 29 days or not.
    - *isLeap* check if the year is leap. It is the year that divisible by 4, and if the year is divisible by 100, it should also be divisible by 400.
  + *isValidChecksum* returns true if the checksum algorithm is similar as the result returned by *getChecksum*
    - *getChecksum* obtains first 9 digits of the ID then perform the Luhn’s algorithm. The even-placed digits (count from 0) are multiplied by 2 then get the sum of each digit in the result, odd-placed digits are multiplied by 1. We then add each result in every step to the total sum. After that we get 10 minus last digit of the total sum to obtain the checksum digit.
  + If all those 3 methods return true then the ID is valid.
* *IDVerificationWithInformationToString* returns the string to report whether the ID is correct (with genders) or invalid ID with reasons. All the outputs in the example are exported using this method.
* *inputHandlingAndConvertingIntoTheCorrectFormToFurtherProcessingIDCorrectly* returns the String in the form of YYMMDD-XXXX. It supports 4 types of strings:

[4Y/2Y]MMDD(-)XXXX.

Note: since processing 2-digit years was the problem of those developers before Y2K, we shall use the Windows XP default approach when dealing with 2-digit years, or only between 1930 and 2029.

## Exercise 3: Pizza

This exercise requires another input for the pizza type, topping types and descriptions.

* Constructor: this constructor has 3 parameters, pizza size, topping type and number of toppings per pizza.
* *calcCost* returns the pizza costs base on the size and cost per toppings for each of the respected size.
* *getDescription* prints out order information.
* *getType* returns type of topping, if there is any topping that is larger than 0.
* *getTopping* returns the number of toppings per pizza.

In the class PizzaMain, it is simply an implementation of the Pizza with more texts.

## Exercise 4: Money

Like exercise 1 and 3, this

* Constructors: there are three constructors:
  + With dollar and cent: set the number of dollar and cent according to the input
    - If the cent is larger than 100 or smaller than 0, the remainder is then taken modulo
  + With dollar only: the cent is set to 0 and dollar to the user input.
  + Without dollar and cent: the default money $0.00 is constructed.

# Source code

Here is my source code for all the exercise.

|  |
| --- |
| 1 |
|  |
| 2 |
|  |
| 3 |
|  |
| 4 |
|  |

# Bibliography

|  |  |
| --- | --- |
| [1] | My2ndAngelic, “SweID,” 30 Dec 2017. [Online]. Available: https://raw.githubusercontent.com/My2ndAngelic/1DV506/master/src/main/java/eh223im\_assign3/SweID.java. [Accessed 1 July 2020]. |