Systems Programming

Part 1:
Object-oriented programming + input/output (I/O) + testing

(Additional development)

1. Introduction

The objective of this additional development related to the first part of the project is to put into practice the knowledge learned about testing in a specific part of your code. Specifically, you will be working with the class `Person` and one of its specializations, the class `Customer` (new class to be developed). You can develop this optional part of the project independently from the mandatory part of the project.

2. Class Modelling

The following two classes are involved in this optional development:

- Person (`Person`). It refers to any relevant person for the management of a warehouse. It can be a customer of the warehouse, the employee that manages the order, or the contact person for a provider. We cannot create objects from this generic class, although we will create objects from the derived classes. A person has an identification number (`id`), first name (`firstName`), last name (`lastName`) and contact email (`email`). You need to create the following methods for this class:
  - Two constructors: An empty constructor and a constructor that receives as parameters all the necessary values to initialize the attributes of this class. The email provided must contain an "@"; otherwise, an `EmailException` should be thrown. You should program the class `EmailException` extending from class `Exception`, which is already provided by Java.
  - Access/Modification methods (`get/set`) for all the attributes of this class. The modification method for the contact email should consider the above-mentioned condition of containing an "@" as well.
  - An additional modification method (`set`) method `public void set(String[] data)`. This method receives as a parameter an array of strings with each of the object attributes, converts these strings to the appropriate data type and calls the corresponding set method to store the values in the attributes. If the number of received parameters is not the one expected, then an `IllegalArgumentException` must be thrown. This exception is already programmed.
  - A `toString` method which overrides the `public String toString()` method from class Object. This method must return a String with all the attributes in the indicated order separated by the character "|" (i.e. "`id|firstName|LastName|email`").
  - The methods to read and write from/to file and from/to standard input/output, which must be developed in the mandatory part of this project, do not need to be tested here (and therefore you can just comment them to complete this optional development).
- Customer (`Customer`). It refers to a customer of the warehouse and is a specialization of a person. We can create objects of this class. Customers have a loyalty card, which has a number that must be unique (`loyaltyCardNumber`), a number of points (`loyaltyCardPoints`), and a frequent status (`isLoyaltyCardStatusFrequent`) which will be granted if the number of points is equals to 100 or above. You need to create the following methods for this class:
  - Two constructors: An empty constructor and a constructor that receives as parameters all the necessary values to initialize the attributes of this class (be aware that the loyalty card number must be unique and is assigned at the moment of creating the object). The number of points cannot be negative; otherwise, a `NegativeNumberException` should be thrown. You

should program the class `NegativeNumberException` extending from class `Exception`, which is already provided by Java.

- Access/Modification methods (`get/set`) for all the attributes of this class. The modification method for the number of points should consider the above-mentioned condition of not being negative, and also that 100 of more points means that the frequent status is achieved.
- An additional modification method (`set`) method `public void set(String[] data)`. This method receives as a parameter an array of strings with each of the object attributes, converts these strings to the appropriate data type and calls the corresponding set method to store the values in the attributes. If the number of received parameters is not the one expected, then an `IllegalArgumentException` must be thrown. This exception is already programmed.
- A `toString` method which overrides the `public String toString()` method from class Object. This method must return a String with all the attributes in the indicated order separated by the character "|" (i.e. `"id|firstName|LastName|email|points|status"`).
- The methods to read and write from/to file and from/to standard input/output, which must be developed in the mandatory part of this project, do not need to be tested here (and therefore you can just comment them to complete this optional development).

3. Testing

You are asked to develop a test class (with JUnit 5) to test the `Customer` and `Person` classes. This test class is expected to achieve **100% coverage of methods, lines and branches in the `Customer` and `Person` classes**. Note that you cannot create objects of class `Person` (as indicated in section 2), so you will need to test this class by creating objects of class `Customer`. Some indications which may be useful to complete this task.

- Create each test using "@Test" notation in JUnit 5.
- Use methods `assertEquals(…)` and `assertThrows(…)` (in the case of testing that an exception is thrown) in the test.
- Try to identify the minimum number of tests that need to be run to achieve 100% coverage.