Exam: Object-Oriented Software Development

January 2022

Name: Student number:

The total duration of this exam is three hours. We recommend that you do not spend more than 40 minutes on the first part of the exam (modelling), and then move on to the second part (Java implementation). You are free to move between parts as you wish.

1. Modelling (20% of your total score, draw your diagram in the dedicated area)

< UML task omitted for mock exam. >

Name: Student number:

Draw your diagram here

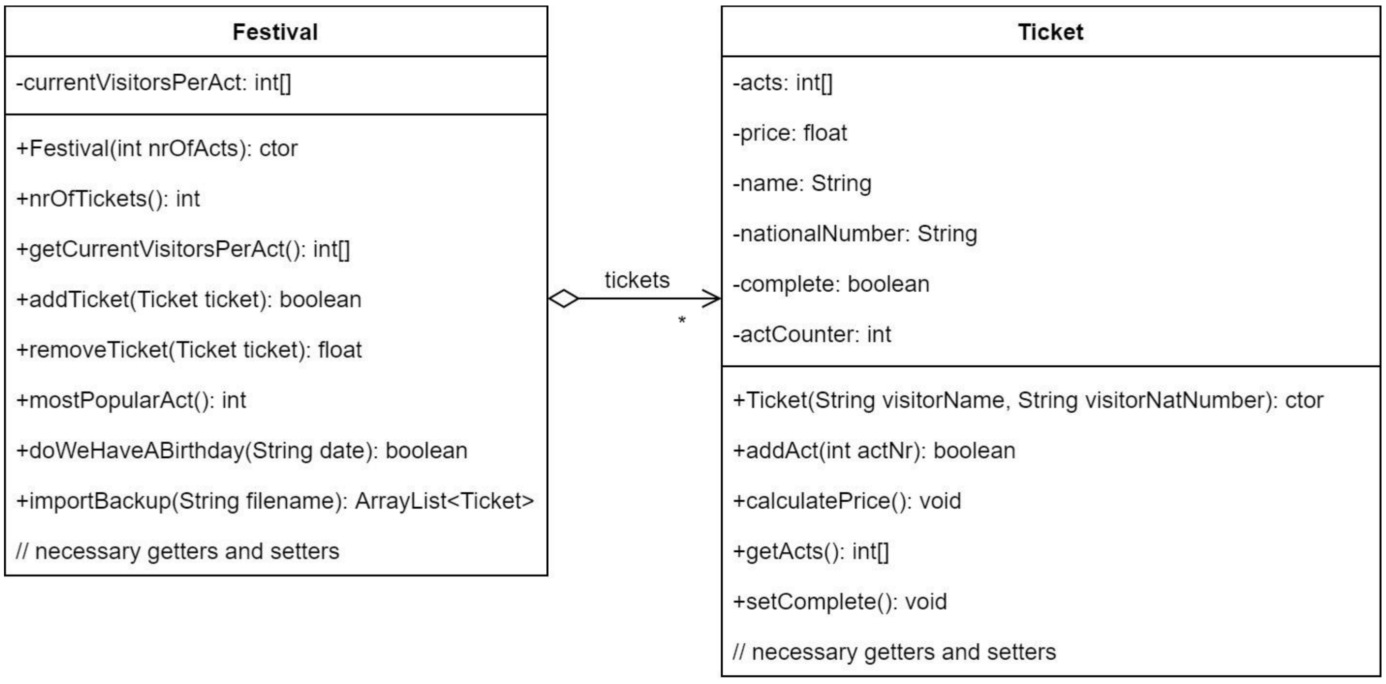
Name: Student number:

2. Java Implementation (80% of total score)

To start the exam, download the *.zip file that is included with this assignment description. The password to unzip will be provided by the invigilator. It is best to remove the original zip file after you have unzipped it. Besides the assignment, there is also a Tools item on Toledo. Here you can find a file with a subset of the Java documentation that might be helpful.

PLEASE BE AWARE: At the end of the examination, you will need to create a zip file of your solution, which you then have to upload to Toledo. Please ensure that you start this procedure early enough to be able to upload your file by the time the exam officially ends. It is your responsibility to create the zip file, ensure that it contains the correct files (and the correct files only), and that it is uploaded on time. **To save your progress, you can also make multiple uploads throughout the exam – in this case, your final attempt will be graded.**

In total, **this exam includes 11 tests** that you need to address, all of which are implementation tasks with respect to the UML diagram included below. You start from the following class diagram, which describes a system to book festival tickets. Please note that **you may not adapt the signature of the given methods, and types of attributes may not be changed** either. It is possible that you will need to add extra classes, attributes or methods in your implementation. **You also must not change the test code in any way.**


Festival
-currentVisitorsPerAct: int[]
+Festival(int nrOfActs): ctor
+nrOfTickets(): int
+getCurrentVisitorsPerAct(): int[]
+addTicket(Ticket ticket): boolean
+removeTicket(Ticket ticket): float
+mostPopularAct(): int
+doWeHaveABirthday(String date): boolean
+importBackup(String filename): ArrayList<Ticket>
// necessary getters and setters
tickets
*
Ticket
-acts: int[]
-price: float
-name: String
-nationalNumber: String
-complete: boolean
-actCounter: int
+Ticket(String visitorName, String visitorNatNumber): ctor
+addAct(int actNr): boolean
+calculatePrice(): void
+getActs(): int[]
+setComplete(): void
// necessary getters and setters

The organizers of a festival want to provide the optimal experience for all visitors, and for this, they have established a system by which you already choose the acts you wish to see upon purchase of your ticket. The booking system needs to distinguish between two types of tickets, standard and VIP tickets. With a ticket, you can select up to ten acts. Additionally, there are differences in pricing structure: at an additional expense, VIP ticket holders can book meetand-greet events. When a standard ticket is purchased, visitors need to provide their last name and their national number, which is kept in the format “yyyymmdd-nr”, for example, “19990101-45” for a person born on January 1st in 1999. Tickets that are complete (i.e., where visitors have completed the selection of acts) are registered with the festival system. Please note that visitors should also be able to cancel tickets and receive the appropriate reimbursement, and that there is additional functionality required as described in the tests below.

TESTS TO BE IMPLEMENTED

[1 point] Test 1 checks the constructor and getters and setters of Ticket. You need room to store ten acts, the initial ticket price before the addition of acts is 0, and the *complete* attribute is false.

[3 points] Test 2 checks if the methods addAct() and calculatePrice() are correct. addAct() should store a maximum of ten acts, with each act being represented by an integer (you may assume any integer number can be an act number). After adding an act, the price is updated. The pricing policy for a standard ticket is as follows: 30€ for the first three acts combined, 7,50€ per act for acts four and five, and an additional 5€ for each further act. If you add a total of ten acts, the complete flag is automatically set to true. If a visitor does not want to schedule 10 acts, he or she can manually complete their ticket using the setComplete() method. Once a ticket is complete, you can no longer add further acts.

[1 point] Test 3 checks the constructor and basic functionality of Festival. Every festival has a parameter defining the number of acts that are organized. This number is used to initialize the array currentVisitorsPerAct.

[1 points] Test 4 tries to add tickets to Festival via addTicket(). The information in currentVisitorsPerAct also needs to be updated, in this case, without any restrictions. The method returns whether the ticket has been added, i.e. true in this test.

[1 point] Test 5 is comparable to Test 4, but checks the restriction that "incomplete" tickets are rejected.

[1 point] Test 6 adds the restriction that you cannot add a ticket having the same national number as already booked onto the festival, ensuring that one person can only book one ticket.

[3 points] Test 7 checks the method removeTicket(), which removes a ticket from the festival (if it was added before). The method returns the amount of money the owner is reimbursed with. For a standard tickets, this is 20€. If you try to remove a ticket that does not exist, the method returns 0. Do not forget to update also currentVisitorsPerAct.

[2 points] Test 8 checks mostPopularAct(). This method returns the number of the act that has the highest numbers of visitors. If multiple acts are found with the same number of visitors, the method should return the first one that is found. If the festival has no visitors yet, the method returns -1.

[2 points] In Test 9, you need to import standard tickets from a text file ("visitors.txt"), using the importBackup() method. Data in the text file has the following format:

- Name
- National number
- Chosen acts, one per line; "-1" indicates that an entry is finished and no further acts will follow.

The method returns an ArrayList of Ticket containing the tickets that were read from file.

[1 point] In Test 10, it is checked whether there is at least one ticket owner celebrating their birthday on a given date. The method returns a Boolean value.

PLEASE NOTE: For the following test, you have to refactor your code to be able to distinguish between tickets and VIP tickets, with VIP ticket extending a ticket (use of *inheritance*).

[1 points] Test 11 checks if you can create a VIPTicket that is a subtype of Ticket. When creating a new VIP ticket, visitors need to decide whether they want to participate in a meet-and-greet with the acts at the festival, which results in an additional fee of 50€ that is added to their ticket price.

At the end of the exam…

When ready, upload your results to Toledo. You will need to make one upload using the upload folder labelled with today’s date. Zip the **ENTIRE** folder that contains your project, and attach the file to the Toledo upload. **Make sure to upload the most recent version of your project – uploading the original file will result in a score of 0 for this part of the exam. Don’t include the Javadocs tool in your upload.**

A last step with respect to the computer-based exam is to click on Exit exam, where you declare that you have uploaded everything you wanted to upload. After this you step, you no longer have access to the exam. **Finally, please do not forget to hand in these sheets to the invigilator.**

IT IS YOUR RESPONSIBILITY TO ENSURE YOU UPLOADED THE CORRECT FILES, AND TO MAKE SURE THIS HAPPENS BEFORE THE END OF THE EXAMINATION PERIOD. THE INVIGILATORS CANNOT CONFIRM TO YOU WHETHER YOUR UPLOAD IS CORRECT.

EXTRA SPACE FOR DRAFTING AND TAKING NOTES – THESE SHEETS WILL NOT BE GRADED

EXTRA SPACE FOR DRAFTING AND TAKING NOTES – THESE SHEETS WILL NOT BE GRADED