Software Developer Course Assessment

Quantitative Assessment Practice

Course Name: Advanced Programming (Java)
Current Week: 13th September 2024
Submission date: 24th September 2024

Introduction:

The purpose of this assessment is to help us understand how the class is doing in terms of the review material that we have covered during the previous couple of weeks. The **only** purpose of this assessment is for us to improve our approach to review and ensure that what we're currently doing is an effective strategy. Completion of this assessment is **mandatory - if you don't submit a solution, it will be marked as incomplete. You must complete a minimum of 80% of your assigned QAPs per course – otherwise you will be marked as incomplete for that course no matter how good your other grades are.** If you do submit a solution, it will be marked as complete, as you will receive full marks no matter what your actual performance was – again this is a participation grade.

Again, the goal here is to help you all in the best way that we can, so please do be honest when answering the questions related to how long it took, which resources you used, etc. And please ensure that you do your **own** work – don't just copy off a friend to get it done, earnestly do your best with it. If you can't get it completely working, give us what you have. While it will be graded, the grade will not count against you, it's just a way for us to see where everybody is, and to know which concepts, if any, we, as a class, may be struggling with.

Deadline: You will have until the end of the day on 24th Sep 2024 to submit your assessment solutions. Please ensure you answer all the questions outlined in the instructions portion of this document as well in your submission.

Marking: In this program core evaluation is marked with one of three possible marks: *Incomplete, Pass, Pass Outstanding.* For QAPs, though, where incomplete marks are more important for our own information as well as for the information of the student, we wanted to increase the resolution of our grading system. Therefore, QAPs are marked on a scale of 1-5. The details of this marking system are summarized in the table below.

Grade	Meaning
1	*Incomplete.* Student shows severe lack of understanding of the material – solution is heavily incomplete, non-functional, or completely off base of what the assignment was asking for.
2	*Partially Complete.* Students show some understanding of the material. Solution may be non-functional or partially functional, but the approach is correct, albeit with some major bugs or missing features.
3	*Mostly Complete.* Student demonstrates understanding of the major ideas of the assignment. Solution is mostly working, albeit with a few small bugs or significant edge cases which were not considered. Shows a good understanding of the correct approach, and is either nearly a feature-complete solution, or is a feature-complete solution with some bugs.
4	*Complete (Equivalent to: Pass.)* Student shows complete understanding of assigned work and implemented all necessary features. Any bugs that are present are insignificant (for example aesthetic bugs when testing the functionality of code) and do not impact the core functionality in a significant way. All necessary objectives for the assignment are completed, and the student has delivered something roughly equivalent to the canonical solution in terms of features and approach.
5	*Complete with Distinction (Equivalent to: Pass Outstanding)* The student demonstrates a clear mastery of the subject matter tested by the QAP. The solution goes above and beyond in some way, makes improvements on the canonical solution, or otherwise demonstrates the student's mastery of the subject matter in some way. A solution in this category would consider all reasonable edge cases and implement more than the necessary functionality required by the assignment.

Instructions:

You are allowed to complete the assessment problems below in whatever way you can but please answer the following questions/points as part of your submission:

1. How many hours did it take you to complete this assessment? (Please keep try to keep track of how many hours you have spent working on each individual part of this assessment as best you can - an estimation is fine; we just want a rough idea.)

2. What online resources you have used? (My lectures, YouTube, Stack overflow etc.)

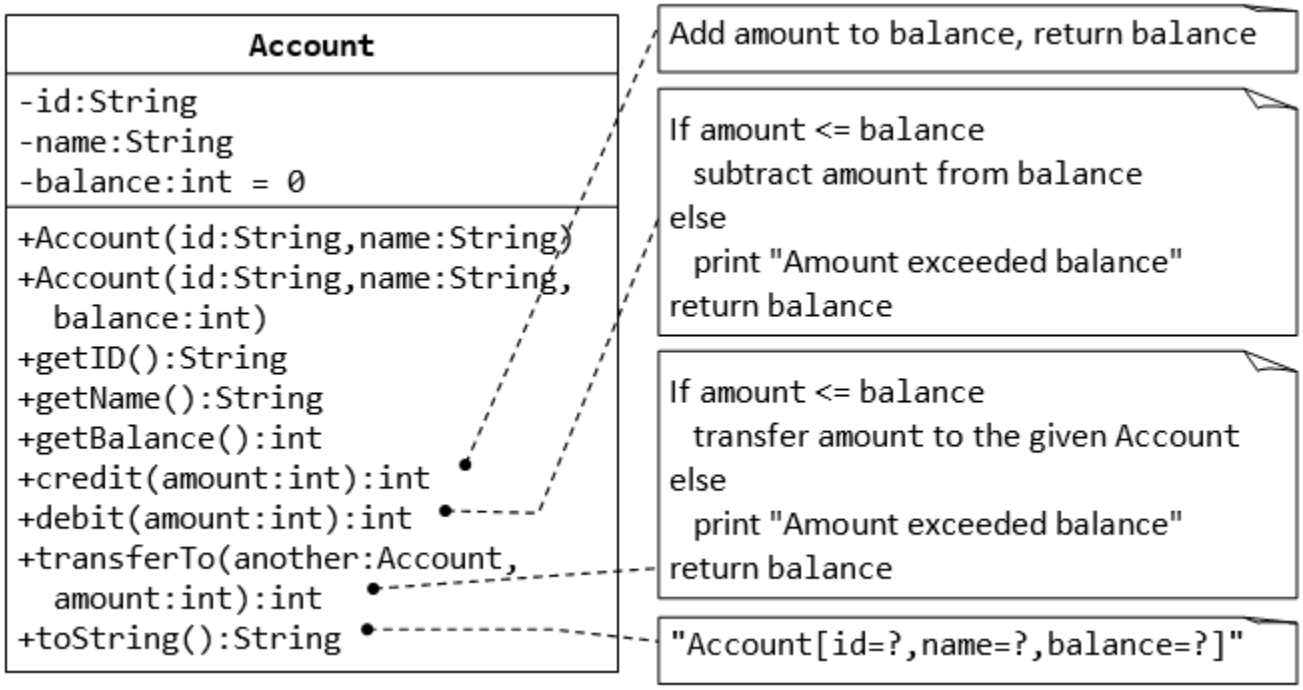
3. Did you need to ask any of your friends in solving the problems. (If yes, please mention name of the friend. They must be amongst your class fellows.)

4. Did you need to ask questions to any of your instructors? If so, how many questions did you ask (or how many help sessions did you require)?

5. Rate (subjectively) the difficulty of each question from your own perspective, and whether you feel confident that you can solve a similar but different problem requiring some of the same techniques in the future now that you've completed this one.

Problem#1:

The `Account` Class



1. Write java code for the Account class explained above. (Account.java)
2. Write a *test program* called TestAccount (in another source file called TestAccount.java) which uses the Account class.
3. Create two accounts
    a. Acc1 with balance of $5000 and Acc2 with balance of $4000 by using parameterised constructors defined in Account class.
    b. Display balance of both accounts using getBalance() method.
    c. Transfer $1000.00 from account 1 to account 2.
    d. Display balance of both accounts using getBalance() method again.

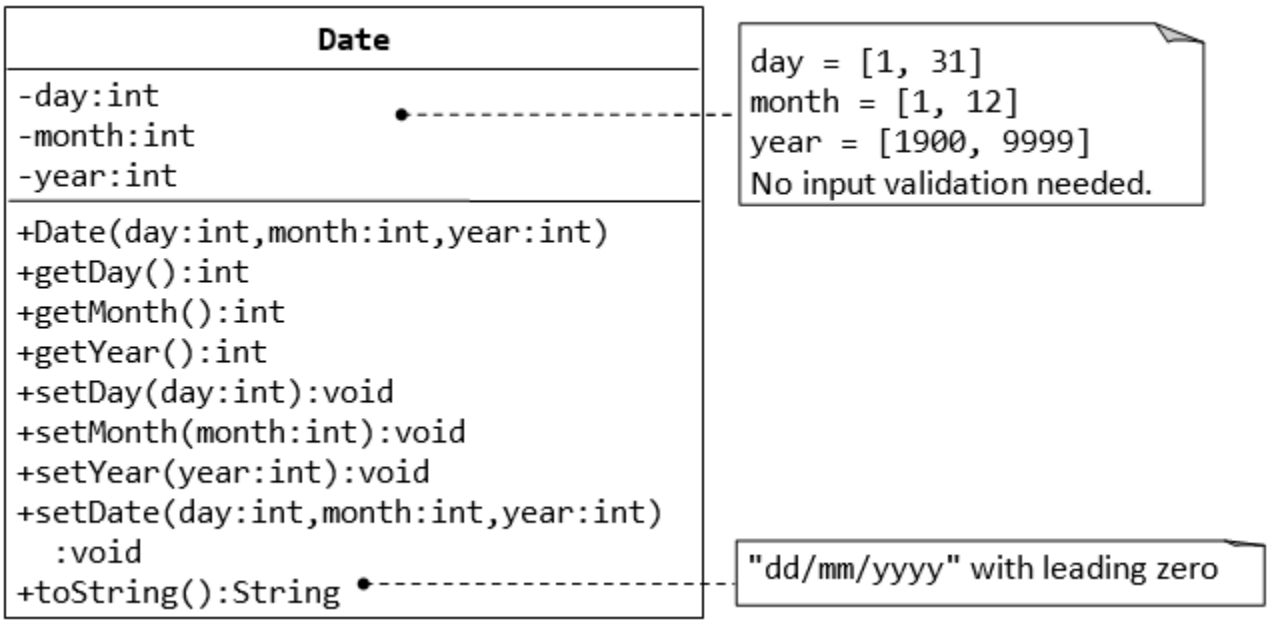
Deliverable#1:

Two complete and working-class files with proper comments.

1. *Account.java*
2. *TestAccount.java*
3. *Screenshots of the running code's output*

Problem#2:

The Date Class



1. Write java code for the Date class explained above. (Date.java)
2. Write a *test program* called TestDate (in another source file called TestDate.java) which uses the Date class.
3. Create a Date object and print it using the toString() method in the format specified.

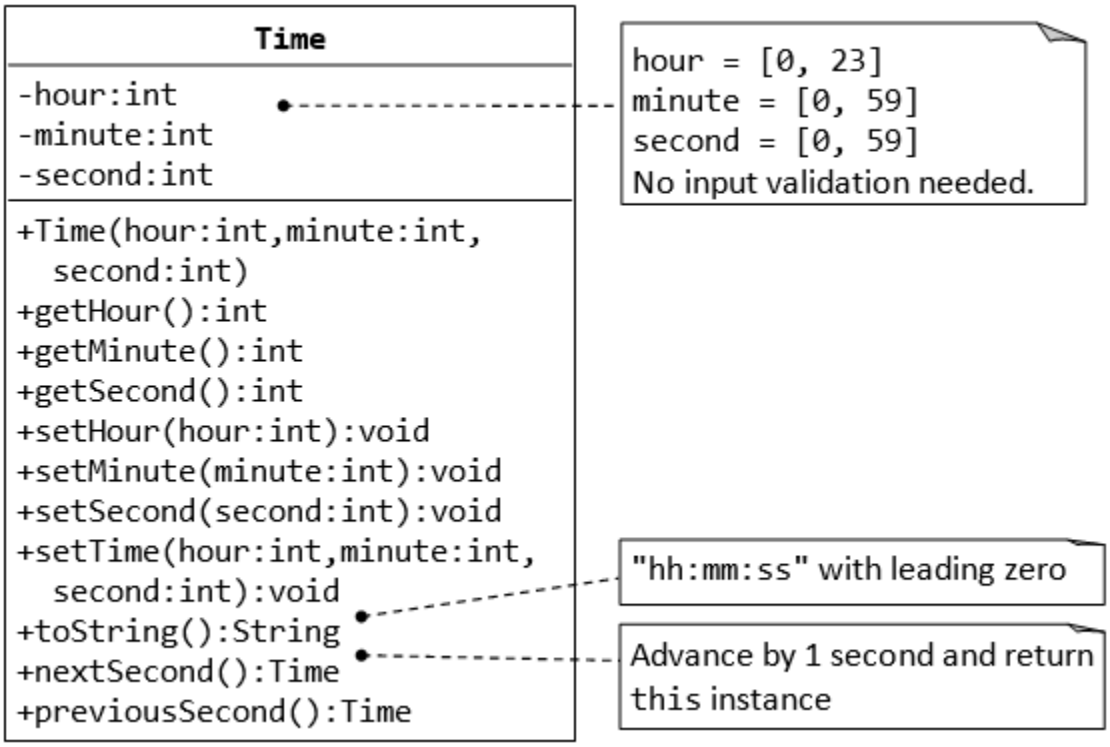
Deliverable#2:

Two complete and working-class files with proper comments.

1. *Date.java*
2. *TestDate.java*
3. *Screenshot of the running code's output*

Problem#3:

The `Time` Class



1. Write java code for the Time class explained above. (Time.java)
2. Write a *test program* called TestTime (in another source file called TestTime.java) which uses the Time class.
3. Create a two time objects (t1 and t2). Set their time to 21:10:15 and 10:20:25 using set methods.
4. Call nextSecond() for t1 and previousSecond() for t2.
5. Display the final times for t1 and t2 using toString() method.

Deliverable#3:

Two complete and working-class files with proper comments.

1. *Time.java*
2. *TestTime.java*

3. *Screenshot of the running code's output*

Submission:

Please create a public github repository and upload all the java files files (Account, Date, Time, TestAccount, TestDate, TestTime), and a word document with the screen shots of all the outputs and their running codes and also the feedback questions. Finally submit the link to that repository.