CPS3222 Assignment

Instructions

- This a group (pairs) assignment. This assignment will be marked out of 100 and accounts for 60% of the final CPS3222 grade.
- Students will be marked individually based on their contribution to the assignment, as verified during the interview.
- While it is strongly recommended that you start working on the tasks as soon as related material is covered in class, the firm submission deadline is Monday 9th January 2017 at noon. Hard copies are required to be handed in to Vanessa Marie Borg, the departmental secretary of the Department of Computer Science.
- You are to allocate approximately 50 to 60 hours of work to this assignment.
- A soft-copy of the report and all related files (including code) must be uploaded to the VLE by the indicated deadline. All files must be archived into a single .zip file. It is the student's responsibility to ensure that the uploaded zip file and all contents are valid. Since this is a group assignment, all members of the group should upload the assignment.
- Reports (and code) that are difficult to follow due to low quality in writing-style/organization/presentations will be penalized.

Additional Instructions

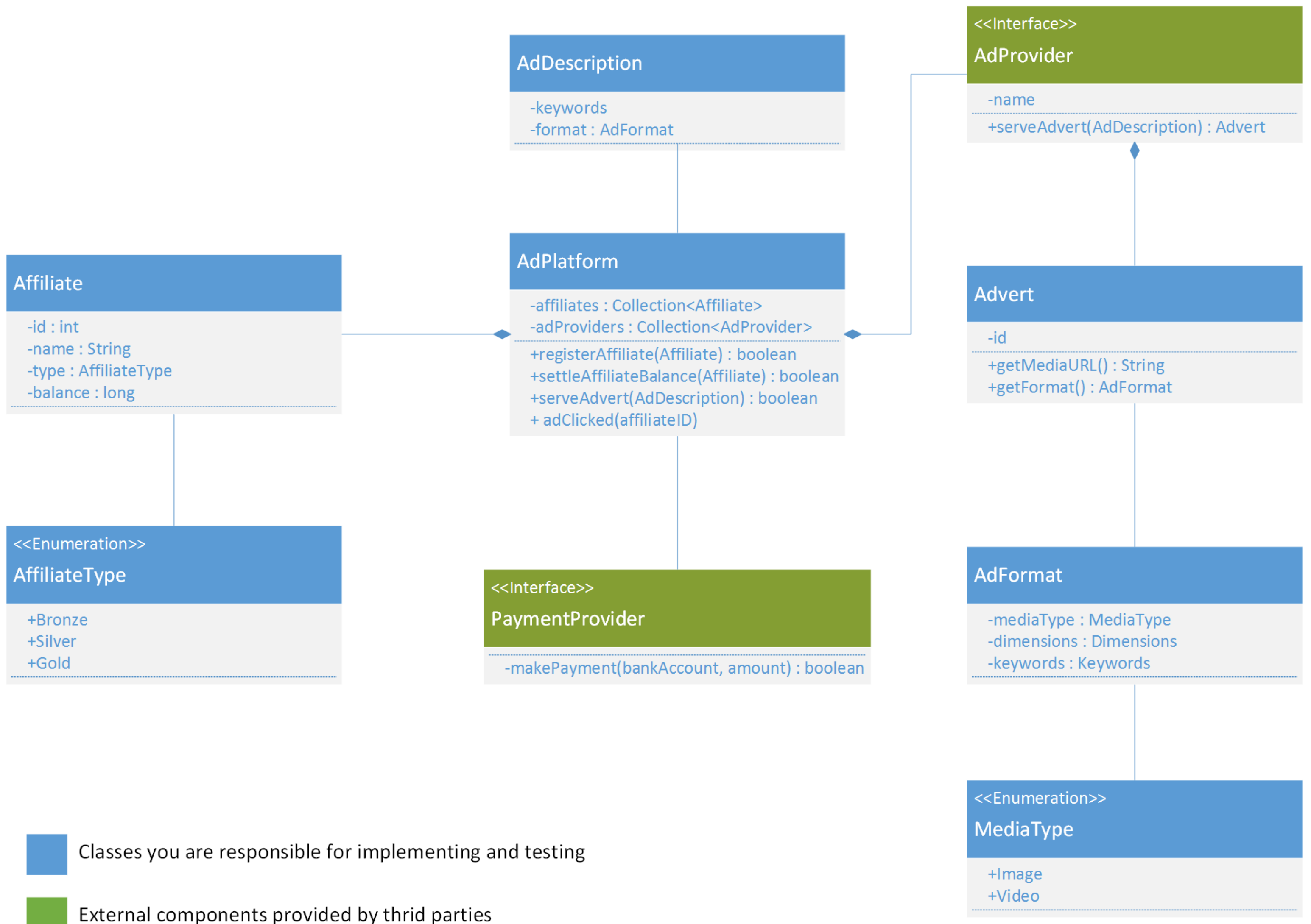
- Work on the assignment is to be carried out by both people at the same time using pair programming on the same machine.
- Please organize yourself in pairs and send an e-mail to mark.micallef@um.edu.mt indicating the members of your group by Monday 14th November 2016.
- You will be required to produce a short report about the assignment and will be asked to defend your work in an interview.

Overview

You have joined a startup that is developing an online advertising platform. The platform links affiliates to advert providers such that affiliates can request adverts to display to users who visit their websites. If users click on an advert then the affiliate will be paid €0.50. Following initial discussions, the system architect has create a first draft of a Class Diagram for the system. This is depicted in Figure 1.

In addition, you have also been provided with the following specifications/rules:

1. Affiliates can register with the system and are assigned a unique ID.
2. New affiliates are assigned an affiliate type of type ***Bronze***
3. When affiliates reach a balance of €50 for the first time, they are promoted to type ***Silver***.
4. When affiliates reach a cumulative balance of €500 for the first time, they are promoted to type ***Gold***.
5. An affiliate can request that his/her balance be settled. This can be done only if the balance is €5 or greater. When this happens, the AdPlatform will ask a payment provider to pay the affiliate 100% less a commission which is to be paid to the AdPlatform account. The commission is 10% for Bronze affiliates, 7.5% for Silver affiliates and 5% for Gold affiliates.
6. Affiliates request adverts from the platform by creating AdDescription objects to specify the type of advertising they need. The platform subsequently polls its list of AdProviders to see if they have any matching adverts and chooses one at random to pass on to the affiliate in the form of an Advert object.
7. The platform is responsible for checking that the advert returned by the provider is in line with the ad format requested by the affiliate. If not, it will return another advert from a different provider or, if no valid ad exists, return a null object to indicate that no ad could be served.
8. When users click on displayed adverts, AdProviders will notify the platform by calling its adClicked() method and provide an affiliate ID. At this point, the platform will update the affiliate’s balance.



🟦 Classes you are responsible for implementing and testing

🟩 External components provided by thrid parties

Figure 1 - Draft class diagram for the system

Task 1: Unit Testing and Test Driven Development (28%)

You are required to build and unit-test the system outlined above using a test driven approach and ensuring maximum test coverage. Document your statement coverage analysis, any test patterns you use, your use of test doubles and any aspects of interest when it comes to coding for testability. Please note that at this stage, you are not required to produce any implementations of the `PaymentProvider` and `AdProvider` interfaces.

The design presented above is a first draft by the architect and may need adjustments in order to function and be fully testable. Feel free to make the design your own but please document any assumptions you make in your documentation.

Task 2: Cucumber and Automated Web Testing (28%)

Develop a web application that uses your classes that supports two pages:

1. A login page for affiliates
2. An account page for affiliates

Your web application should behave according to the scenarios below. Document any test doubles you needed to implement as well as any issues relating to test data that you came across.

Scenario 1: *Successful Affiliate Login*

```
Given I am an affiliate trying to log in
When I login using valid credentials
Then I should be taken to my account admin page
```

Scenario 2: *Unsuccessful Affiliate Login*

```
Given I am an affiliate trying to log in
When I login using invalid credentials
Then I should see an error message
And I should remain on the login page
```

Scenario 3: *Account Admin Page Contents*

```
Given I am a logged in affiliate
When I visit my account admin page
Then I should see my balance
And I should see a button allowing me to withdraw my balance
```

Scenario 4: *Withdrawals*

```
Given I am a logged in affiliate
And my balance is <balance>
```

```
When I try to withdraw my balance
Then I should see a message indicating <message-type>
And my new balance will be <new-balance>

Examples:

|balance|message-type|new-balance|
|4.99   |error       |4.99       |
|5.00   |success     |0.00       |
|0.00   |error       |0.00       |
|142.12 |success     |0.00       |
```

Task 3: Model Based Testing (24%)

Modify your web application to support simple ad delivery from one AdProvider implementation. Your stakeholders are now interested in generating test cases based on the following rules:

1. An affiliate initially has a balance of €0.00 and an affiliate-type of Bronze
2. Every time an affiliate requests an ad and a user clicks on it, the affiliate's balance will be increased by €0.50
3. When an affiliate generates his first €50 then he is promoted to an affiliate-type of Silver. Note that even though an affiliate's balance might drop due to withdrawals, a running total of the amount of commission earned by an affiliate should be tracked.
4. When an affiliate generates his first €500 then he is promoted to an affiliate type of Gold. Note that even though balance might drop due to withdrawals, a running total of the amount of commission earned by an affiliate should be tracked.

Create a model which describes how this aspect of the web application should function. Use this model to automate the generation and execution of test cases for this aspect of the system. Document the following:

1. Your model in a graphical notation
2. A graph depicting how much state coverage was achieved against the number of test cases executed.
3. Explain what step(s) you would take to reduce the amount of test cases required to achieve 100% state coverage.

Task 4: Performance Testing (20%)

It is expected serving and clicking of adverts will generate heavy usage loads on the system. Stakeholders are concerned about the extent to which the serveAdvert() and adClicked() methods can deal with large usage loads. Using a performance testing tool of your choice or a custom-built test harness, provide an indication of the number of affiliates that your system can handle. Each affiliate is expected to request between an everage of 2.5 adverts per second with end users expected to click on 10% of delivered adverts.