Assignment 5a/b - Design Patterns

Version: June 10, 2021

General

In this assignment, we will introduce Design Patterns while also incorporating everything you've learned so far by creating a code-a-mon game simulator.

This document covers everything you need to do, but keep in mind that this assignment is big and is designed for 2 weeks overall. THIS ASSIGNMENT WILL TAKE TIME!

You should apply all you have learned so far, testing, good GitHub workflow, commits, clean repo etc. It will not be mentioned specifically it is implied.

You will need to create a PDF document (specifics below) for your Deliverable B which you will submit on Canvas.

Make sure you stick to the requirements in this assignment and include all necessary information in your PDF. We will not reward points later on if you forget to submit something (this includes the PDF). Make sure you clone your repo at the end and test if it runs to make sure that you have put all necessary files in the repo. This is good practice anyway so you are sure what you submit actually works.

This assignment is split up into 2 deliverables (a/b) to force you to work continuously and not just in the last couple of days. See below for details on what you need to submit for each deliverable.

The specifics

1. Work on Git and GitHub for this assignment (on a private repo, add ser316asu as collaborator). (7 points)
    - Name your new private repo following this pattern: ser316-<semester><year>-<session>-<asurite>-DP (as an example: ser316-summer2021-C-sparky11-DP).
2. Each Design Pattern below is worth 15 points (45 points) :
    - You need to create 3 design patterns from the Gang of Four for the requirements later in the document.
    - You do need to implement different Design Patterns
    - You have two choices:

   a) The minimal requirement for up to 95% of credit: Have the Design Patterns implemented independently from each other (the patterns should not "work together"). Basically, each one of them can be run separately. In this case, you would implement each pattern in a separate package. In a Main class, first show the functionality of one pattern and then the other etc. This is the easier version, since the DP are not intertwined which many struggle with.

   b) Getting full points plus some extra credit: Implement your code with the Design Patterns being intertwined (you can of course still have packages if you like), working together and forming one good project. The DP still need to be implemented correctly and make sense!

- Make sure you comment your code well so we can see what you are doing or tried to do.
- You do NOT need to meet all the functional requirements specified later, but each Design Pattern should at least implement 3 of the requirements (not all might fit into your Design Pattern but might just be other functionalities in your code). In your PDF, describe which requirements you fulfilled and also make sure you comment on it in your code. If you chose version 2 above, then 9 requirements need to be fulfilled overall in your code.
- The code needs to be fully functional and needs to "do something"! There needs to be a basic simulation that shows the functionality of the requirements you chose. Just creating the skeleton of the Design Pattern with no data is NOT sufficient. I want to see a running functional implementation.
- Make sure your Main does have good print outs so we and you can see what is going on.
- We will grade you based on the code that you've written. This is based on all you have learned so far: Good coding practice, testing, coding standards, clean repo, etc.
- Minimal Requirement for each package: There is some kind of simulation included in Main that includes a world, trainers, code-a-mons (so you need to instantiate some of these and they can be hard coded or random), some modifiers like weather and/or type advantages. The simulation ends with some kind of well chosen end scenario, e.g. all but one trainer's code-a-mons faint, a trainer has X victories, or a trainer defeats a final boss.

3. Create a Gradle file (5 points) - requirements (**All the below Gradle requirements are mandatory, you will not get points for this assignment if this does not work**). We will NOT import anything into an IDE!
   - Your code HAS TO execute via "gradle run" via command line
   - Include Checkstyle and Spotbugs
   - Include JUnit 4

4. Checkstyle and Spotbugs: make sure you adhere to the coding standards you set in your Checkstyle XML and that Checkstyle and SpotBugs do not show anything. If you cannot get rid of everything, you should mention this in your PDF document and explain why you were not able to fix it. In your PDF, include a screenshot of your Spotbugs and Checkstyle report. You will not get points for this if there are no screenshots that show us your reports. (10 points)
5. TravisCI: Set up TravisCI so it builds your project, runs your Unit Tests and also Spotbugs and Checkstyle. (5 points)
6. Include JUnit tests (JUnit4 or JUnit5) and test your code. You should reach at least 80% code coverage (excluding your Main, getters and setters). Include a screenshot of your JUnit and Jacoco report, showing your tests pass and your code coverage. You will not get points for this if there are no screenshots that show us your reports.(15 points)
7. Include a short screencast (does not have to have audio but is appreciated) showing you doing a git clone from your repo, gradle build, showing your reports (JUnit, Spotbugs, Checkstyle, TravisCI), and gradle run (max. 2min). (8 points)
8. README.md on GitHub (well written and easy to find everything)
    - Include a link to your screencast
    - Explain each of your Design Patterns briefly and mention which of the requirements you fulfilled with this Pattern. (5 points)
9. You can earn up to 15 points extra credit, see at the end of this assignment for specifics.

IMPORTANT: If your Code has compile errors you will receive 0 points. Make sure your Main runs without errors. Example you are only able to make 2 of the 3 Design Pattern run in your Main and the 3rd is implemented but does not work. Then you should not include number 3 in your Main. You will get credit for the the two DP and will get partial credit for number 3 (if it has no compile errors).

As stated before, you do not have to implement all the requirements and you can choose any 3 design patterns from the Gang of Four for task 1 that you see fit. The HINTS might help you (they are not requirements). You are relatively free on how to do things. Please mark where you see your Design Patterns in your code. Again, this is coding intensive. Start early, as it will take a while to figure things out.

Requirements

Below are some of the functional requirements for the application.

Monster Kampfarena

- A new world must start with a number of trainers, at least 2, each of them starts with at least one code-a-mon. You can also decide to have trainners join in time if you like.

- The trainer can acquire more code-a-mons (max of 6) throughout their adventure. How they acquire them is up to you. An example could be they win one as a reward for every X amount of battle victories or they can catch them.
- Code-a-mons will compete 1v1 with another trainers code-a-mons.
- The simulation should run on cycles. A cycle is considered to be of 2 parts - 1 day time and 1 night time.
- Each cycle should have it's own weather event (these are some examples to give you ideas, use your creativity!):
  - Day 1 - Sunny
  - Night 1 - Clear
  - Day 2 - Rainy
- Weather events should benefit certain types of code-a-mon's stats while being a disadvantage to others (these are some examples to give you ideas, use your creativity!):
  - Sunny: Fire type gains 25% increase in stats and decreases water type by 25%.
  - Clear: Neutral.
  - Rainy: Water type gains 25% increase in stats and decreases fire type by 25%.
- Code-a-mons should be of different types and gain advantages or disadvantages based on their opponent's type (these are some examples to give you ideas, use your creativity!):
  - Water > Fire
  - Fire > Grass
  - Or, you can get creative and change the types from elements to coding languages

    C++>Java

    Java>Python
- Battles with other trainers (or wild code-a-mons are done during the day time). Battles with trainers earn money and experience points while battles with wild code-a-mons only earns experience points.
- During the night these things are possible (choose one or all):
  - Code-a-mons can heal (based on whatever you come up with, maybe potions maybe a percentage, be creative)
  - Can attempt to catch a new code-a-mon (if you chose to catch them) or if they reached the win threshold to receive one, they would get it at this time.
  - Evolutions of code-a-mons can occur.
  - Items can be purchased from the store with money. Items could be potions, items to catch wild code-a-mons with, stat boosters, etc.
- Code-a-mons should have at minimum:

  - Stats: Attack, Defense, Health (Others like speed and so forth can be added if you would like to implement).
  - They should have 1-4 different attacks
  - Each attack has a specific type which gains bonus damage if it matches the type of the user.
  - Attacks should have a chance to critical strike (Double damage).
  - Attacks should have a chance to miss.
  - Attacks should deal a minimum of 1 damage unless they use an attack that deals 0 base damage.
- Code-a-mons gain experience points from winning battles and can level up after earning enough points. Evolutions can occur after reaching certain levels.
- Only one battle can take place at a time. A battle is always between two trainers and each using one code-a-mon. OPTIONAL: You can also decide to have one main trainer and everyone battles against that main trainer instead of all trainers battling with each other.
- During battles with trainers:
  - Each trainer has 1 code-a-mon on the field at a time.
  - Attacks should be performed in a turn-based manner, one code-a-mon attackes while the other defends, then vice-versa until one faints.
  - When a code-a-mon's health reaches 0 or less, they faint, and this particular fight is over. Code-a-mons can heal during the night and be awake again the next day, maybe with less strength or depending on potions etc.
  - When a trainer has no code-a-mon left (no code-a-mons awake during the day), they leave the battlefield
  - Experience points can be handled in two ways: when a code-a-mon defeats another or when the battle is over, it can be given either to the entire team or to just the code-a-mon that won.
  - For a trainer's turn, they can either attack or use an item such as a potion, stat booster, etc.

It is all purposefully pretty wide open to give you more options and have you think about things more. You can be creative on this; if you like a different idea better, go for it. It is more about having a rough framework in which you want to work and doing something fun with implementing Design Patterns than the particular requirements.

Some HINTS (not requirements):

- You could use decorator pattern for evolutions.
- To build new code-a-mon or trainers you could use the factory pattern.
- The simulation should be tick-based. (Mediator pattern). Each tick something should happen, the trainer battles, catches new code-a-mon, purchases items, etc.

Extra Credit more specifics (15 points)

In your document include a new section for the extra credit part, explaining your idea briefly.

For this your Design Pattern shoudl work together (7 extra points if it works together well and uses the Design Patterns well).

The rest of the points can be gained for the following:

The main thing is that you can run your code through an executable jar file which requires a JSON file as input. I want to be able to call 'java -jar asurite_DPextra.jar jsonFile.json' to run your application from command line.

The application should read the JSON file. The JSON file should specify the initial setup, eg. the trainer/s, code-a-mons, etc. (whatever you come up with and is needed for your application). You need to provide at least 2 sample JSON files in your repository.

Now your application runs until either the trainer's code-a-mon/s faint or reaches 100 battle victories or whatever end game you decided on.

Submission A (15 points)

For submission A I want you to have your basic setup done. Private repo, simple Main (can just be a hello world) that can be run through Gradle, Travis CI included (and passing), Checkstyle, Spotbugs, basic test setup. So basically everything but the real implementation should be done.

I also want to have a short summary in the Readme.md on GitHub which explains your rough idea for the Design Patterns you plan to implement (you can still change your mind later).

Submit your link to the private repo on Canvas (ser316asu added as collaborator). No PDF is needed at this point and no screencast.

This submission has 15 of the overall points

Clean Git repo - 3pts
Gradle works correctly - 3pts
Checkstyle and Spotbugs included - 3pts
TravisCI setup and passes - 2pts
JUNIT included - 2pts Readme has basic idea explained - 2pts

Submission B (85 point)

You need to submit your code with your Design Patterns and all that is asked above

1. **A link** to your GitHub repository (also include in the PDF)
2. **A link** to your Travis CI repository (also include in the PDF)

3. **A link** to your screencast (also include in the PDF)
4. **A PDF document** explaining everything that is asked above in the general assignment. Submit the PDF directly on Canvas.