Assignment 2

Part-I

Expectations

Implement a Java program that contains these kinds of smells

- Implementation smells (at least three from the list below)
  - Complex method
  - Long parameter list
  - Magic number
  - Complex conditional
  - Long method
- Design smells (at least three from the list below)
  - Insufficient modularization
  - Multifaceted abstraction
  - Rebellious hierarchy
  - Deficient encapsulation
  - Cyclic hierarchy
- Architecture smells (at least one from the list below)
  - Cyclic dependency
  - Feature concentration

Constraints

- The Java program must not be a dummy program (should not just have some print statements) but a functional program (i.e., must be implementing a (small) problem of your choice).
- Please include a short description (such as location (package/class/method, line no where applicable) of each smell in the readme file along with why you think it is a smell. Also, if you have taken an open-source project to get started, you need to indicate where exactly you changed the project to introduce smells. Remember, you are allowed to take any open-source as the base code, but you must change it to introduce smells (existing smells won't count).
- Multiple instances of a kind of smells will be count as one smell covered.
- It must compile.

Assumptions/Tips

- You may take one of your previous Java projects (even if you developed it for other course) as a starting point and modify it accordingly to introduce smells.
- You may also take any open-source Java project as a starting point. You must declare in the readme file whether you are using your own program or an open-source program.

Delivery

- A Java project is expected to be delivered via Gitlab (place the project code in your Gitlab's assignment2 subproject).
- Ensure that your readme file has all the necessary information needed.
- Paste the link of your assignment repository in Brightspace.

Rubric

- Prerequisite: the program must compile (the evaluation will not proceed without it)
- Each design and implementation smell present in your program will get you 0.5 marks; a total of 3 marks. Architecture smell present in your program will get you 1 mark.
- If it is a dummy program (it just print something random or empty methods), you will lose 1 mark. If the program is simply an open-source project without your changes, you will lose all the marks.

Part-II

Expectations

- Choose a well-known Java open-source project from GitHub.
- Carry out a software design assessment for the project using DesigniteJava. You need to choose at least **ten** different commits spread across the project's life. It is expected that you choose these commits (based on their commit-time) almost equal distance with each other.
- The assessment must include a plot between LOC and the total number of smells (both at Y-axis) versus commits (to see how LOC and the total number of smells are changing from each commit).
- The assessment must include plots between smell density (for each implementation, design, and architecture) smells and corresponding commits. Smell density is computed as follows (for implementation smells):
    - smell density (implementation smells) = (total implementation smells *1000)/total LOC in the project
- The assessment must include plots between commits (X-axis) and metrics (maximum and average metrics for the following metrics: LCOM, WMC, and class LOC) at Y-axis. Please note that you must not include -1 (especially with LCOM) in your max/average computation because it reflects the tool could not compute the metric for that class.
- Identify at least two interesting and relevant observations from the analyzed information. For example, LOC reduced in commit 5 but the total number of smells increased, or, architecture smell density reduced from commit 6 to 7 but implementation smell density increased between these commits.

Constraints

- Each student must work on a unique project. Hence, if you identify a Java project for yourself, make an entry in the excel sheet to claim the project. Obviously, if the project that you want to analyze is already claimed by someone else, you need to find another project.

- Your Java project must have at least 1000 commits and must have at least 50 stars on Github (hint: use GitHub's advanced search or SEART tool)
- You will use DesigniteJava (Enterprise edition) to analyze the selected project.

Assumptions/Tips

- You may write your own script to select the commits and analyze each commit using DesigniteJava but it is not mandatory (i.e., you may do the analysis manually too).
- If your Java project is larger than 50 thousand LOC, then you will need an academic license of DesigniteJava. You may request the license here for free.
- The resources (memory for instance) needed for analysis using DesigniteJava depends on the project size. Run the tool on the latest version to ensure that the tool can analyze the selected project with the available hardware. Also, check out the -Xmx flag to be used while running the tool.
- You may use any tool/software/library to generate plots.

Delivery

- You need to create a presentation with the required information (plots etc.)
- You will upload the presentation on Brightspace before the deadline.

Rubric

- Plot between LOC and the total number of smells against commits = 1
- Plot between smell density (Design/Architecture/Implementation) and commits = 0.5 x 3 = 1.5
- Plot between max and average metrics (LCOM, WMC, class LOC) and commits = 0.5 x 3 = 1.5
- Interesting observations = 0.5 x 2 = 1