A01 Report

Date: 2024-01-03
By: Georgios Dimitrios Samaras(GDS)
Email:georgiosamaras@hotmail.com
Program: Software Development

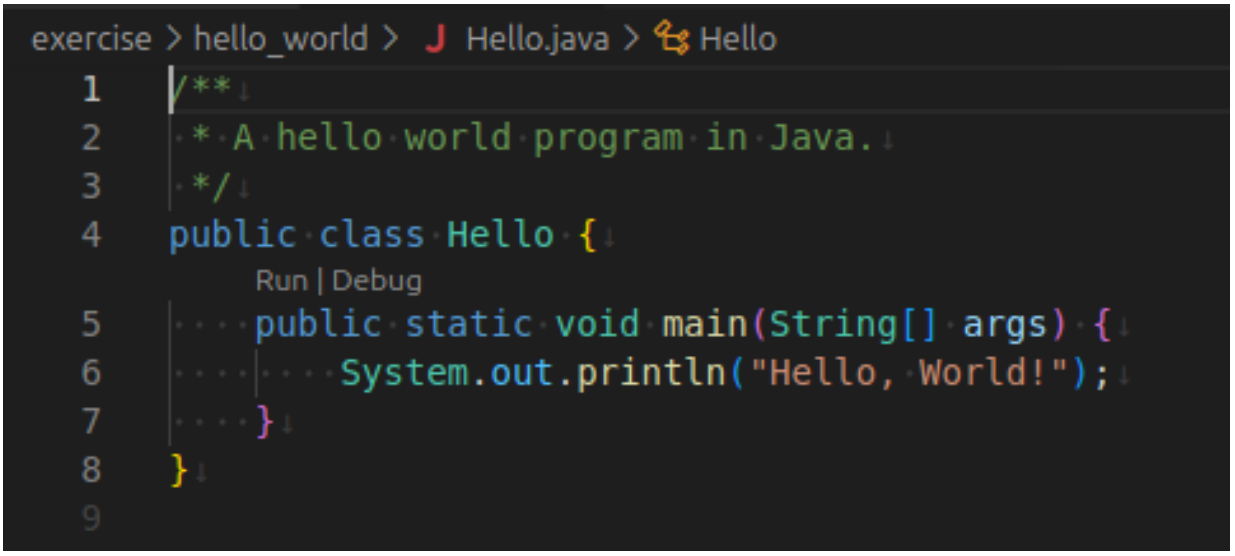


TABLE OF CONTENT

Introduction

Here follows the written report to all assignments in a01/
https://www.dropbox.com/scl/fi/45442p48mpxoj728h0r4c/2025-01-10-16-15-04.mkv?rlkey=4n23cslvxtdga7vwcmxubd3e6&st=mfzfqf8x&dl=0

Task 01

1.Describe your thought process while developing and testing the code for the tasks.
My plan was to try following the instructions to a tee, very slowly getting each component completely done before moving to the next one. And for each component I read the instructions many times before getting the logic and syntax to at least compile, checking with “System.out.println” to see if each method was being called, and then moving to the next.

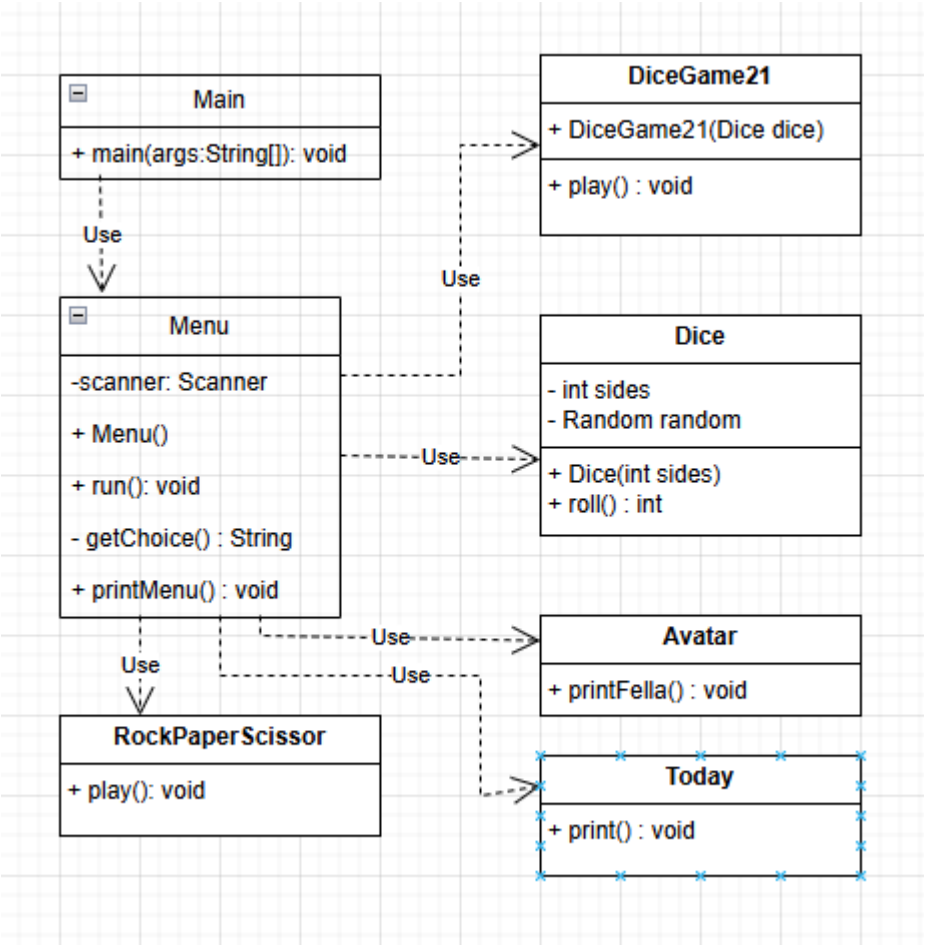
2.Do you find the UML class diagrams helpful when solving the tasks?
They’re definitely helpful, being able to visualize the problem in a more concise way makes it easier to solve when it becomes harder to imagine what the solution is supposed to look like.

3.Elaborate on some of the difficulties learning Java as a new language and writing the first program in Java.
The language requires a lot of boiler-plate, there’s so much syntax and typing involved to create even the smallest bit of code. Getting started with all the classes, files, inheritance and compiling is quite overwhelming too.

4.Describe the code you wrote and talk about how you structured the code.
Menu works as the center of all the codebase, minimal logic in order to call all other classes and their methods.

RockPaperScissors uses a String array where all the options reside, and userChoice will go and get the corresponding option and start the game logic. DiceGame starts the user's and the avatar's roll total, and then Scanner sees if the user wants to roll again, adds the int number to their total, and repeats. Avatar keeps rolling while it's under 17 total, and then both totals are compared and determined with who won the round.

5.Create a UML class diagram to show how you implemented the class(es) and include it in the report.



6.What is your TIL?

Learned quite a bit about objects and methods and how they connect with other files and such. Thinking more about coding in Java instead of trying to code Python-y Java.

Task 02

1. Explain how you structured and implemented the Menu class, could you make it somewhat
independent from the application code and easy to reuse?
The menu class is responsible for most of the functionality in the assignment. It renders all the output for the user, calls other classes, handles input, and sends data around the project.
I started by constructing the menu, printing it, and then managing the user's choice to decide which method to be used. Was a bit hard to understand in the beginning, but i found this way more intuitive after a while. Having most classes serving as "skeletons" for the menu class to build on top of them was fairly nice. With this it should be easy enough to adjust and change things in a reasonable way.

2. Elaborate on how hard/easy it was to follow the class diagram.
This was the hardest part, following the diagram sort of helped making the assignment harder to understand than what it actually is. Pseudo-code and diagrams still leave a gap between the language and what the code should be, and UML doesn't help as much with that. The icons and arrows used to determine what is what are a bit unnecessary, i would much rather have it explicitly say what’s being asked.

3. Do you understand the concept of "dependency injection" and did you use it in your application, if so, explain how you used it?
Dependency injections are a way for classes to allow other objects to use their methods, without the need of hard-coding values in the classes. These classes can be seen as lego blocks, they are shapes which other classes may get and implement as needed.

4. Elaborate on what you find the most challenging parts with this task?
Getting started took a while. Comprehending the problem and the guidance was a bit confusing, and the syntax in Java still feels like dark magic. Keeping a mental track of how inputs are being thrown around all the methods and classes is the hardest. I struggled figuring out how all the files were going to interact with each other.

5. What are your thoughts on writing comments like JavaDocs?
It is a bit verbose, but very useful. Sometimes it takes a while to know what to write in the comments without making it redundant but it is definitively a nice tool to use in developing.

6. Did you do the optional part, if so, elaborate on how you solved each of the optional parts.

7. What is your TIL?

Today I learned how to utilize classes, grab their functionalities, and use them somewhere else in the codebase. Also practiced a fair bit of Java syntax and how to write more "Java-y" Java code.

Task 03

1.How was it to work after a defined class diagram like this, is it helpful or is it hard to follow the thoughts of the designer?

It is helpful up to a point, but diagrams made by someone else add a new layer of abstraction, where I gotta think like the designer did. And for the most part a well designed diagram is what a developer wants to follow anyway, but it is overcomplicated to see so much in one go.

2.Elaborate on the difference between inheritance, composition, interface and abstract classes.

Interfaces are similar to small lego blocks, every class using that interface has to use those blocks as foundation to build on top of. Inheritance is used when many classes share things in common, these classes would have one parent class which they get common functionalities and methods from. Abstract classes are like interfaces, they provide methods for other classes to implement as needed. Composition is a way to achieve inheritance, but in more flexible manner. Using composition allows classes to just be "part" of the inheritance, instead of "being" it. Classes "have" a composition, instead of "are" an inheritance.

3.Do you think the class diagram is balanced for this application, or would you like to propose any modifications to the class diagram?

I think it is balanced, even if a bit too granular, but once everything comes together it all makes sense.

4.Are you satisfied with your code or do you see improvement areas for it?

I'm relieved with my code, it feels very verbose but that's how Java is. Some of the structure is all over the place, reading it from top to bottom can be quite daunting, and the logic in the hunter's movement needs some balancing.

5.What was the hardest part(s) to understand while building this application?

Getting Gson's dependency was a nightmare, gradle just didn't recognize it for some reason and the solution was building everything again. And there were so many layers of abstraction, inheritance and so on that it was hard to keep up. The word position being thrown around 1 billion times felt mind-numbing.

6.Did you do the optional part, if so, describe how it works and elaborate on how you solved it.

I've made a smarter Wolf AI, it moves according to the player's position in order to close the gap. After each player move it checks if its horizontal and vertical position is greater or smaller than the wolf's current position, and then moves right or left, and down or up once each.

7.What is your TIL?

Got much more comfortable visualizing how objects and classes interact, mentally and in code. And feel more capable coding in general, engineering logic and functionalities.

Task 04

Bonus points

Task03/Create a really smart wolf.

Summary

Summarize the most important of your learnings and findings from all the labs to write a summary of how this assignment helped you learn Java.

Elaborate on your thoughts of the Java programming language this far?

TIL;

Learn a lot about research and how to use the lectures and exercises in different applications. And the learning curve it takes to get somewhat useful in a new language, new paradigm, new concepts and techniques is very valuable.

References

[1]Java Interface, BroCode Available: https://www.youtube.com/watch?v=GhslBwrRsnw

[2] Gradle User Manual [Online]. Available: https://docs.gradle.org/current/userguide/userguide.html

[3] Oracle Java Docs. Available: https://docs.oracle.com/javase/tutorial/java/