A01 Report

Date: 2024-11-17
By: Noah Nemhed Uppgren
Email: noahnemhed@hotmail.com ||
noah.nemhed_uppgren0001@stud.hkr.se
Program: Datasystemutvecklingsprogrammet

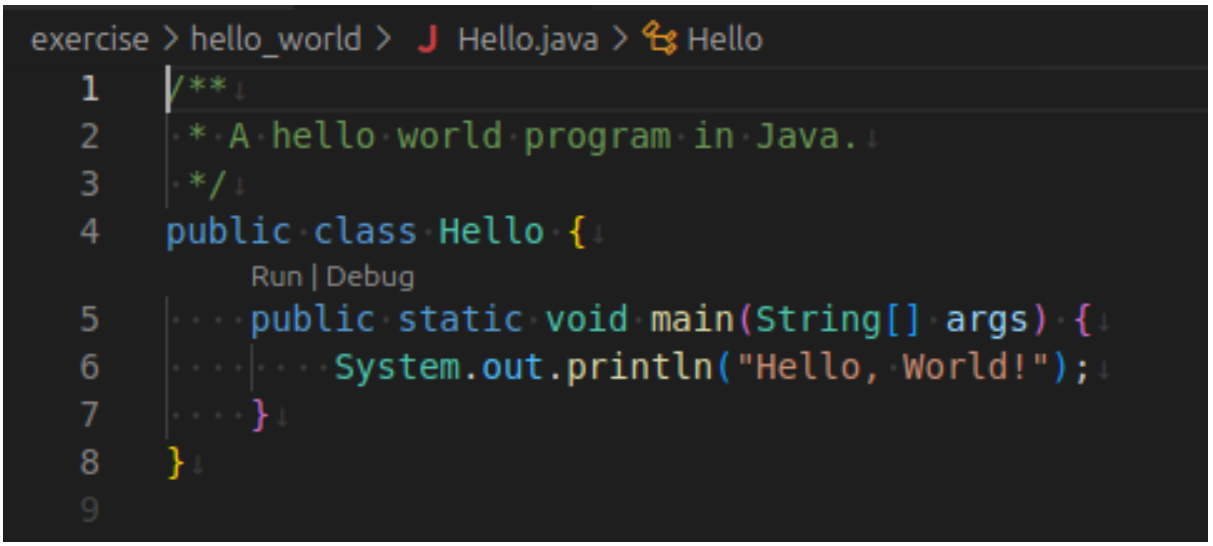


TABLE OF CONTENT

Introduction

Make an introduction to this document, what it contains and why it is written.

Add a link to the recorded presentation, if any.

Task 01 - Get going with Java classes

1. **Describe your thought process while developing and testing the code for the tasks.**

   Jag tyckte vid första anblick att uppgiften kändes svår, men då vi fick koden för menyn så var det inte speciellt svårt att förstå hur man skulle bygga vidare menyn för att inkludera Sten, sax & påse och dice 21 spelet. Att bygga sten, sax & påse spelet var inte speciellt svårt, det var bara att tänka logiskt och använda if-satser för att täcka de möjliga utfallen, dice 21 spelet som jag själv aldrig spelat var lite svårare då jag inte visste lika väl hur spelet spelas. men efter att ha läst instruktionerna så kunde jag ganska enkelt implementera det.

2. **Do you find the UML class diagrams helpful when solving the tasks?**

   UML-klass diagrammen hjälper en väldigt mycket, det hjälper en att förstå hur man ska börja och strukturera själva klasserna för att få en “bas” av hur applikationen ska se ut. När man väl gjort det så blir det enklare att implementera “logiken” för programmet och sy ihop klasserna.

3. **Elaborate on some of the difficulties learning Java as a new language and writing the first program in Java.**

   Att gå från Python till Java är rätt tufft, Java är mycket strängare med syntaxen, då det är ett statiskt typat språk man måste deklarera vilken datatyp varje variabel ska vara i förväg, vilket man inte behöver i Python. Annars tycker jag att de jag lärt mig från python ganska enkelt gick att föra över till Java med lite Googling. Exempelvis skillnaden i For-loopar. I while loopen och if-satserna var det inte så mycket skillnad förutom “måsvingarna” som man inte har i Python. Men efter att ha använt “måsvingarna” ett tag nu så tycker jag att de gör så att koden blir mer läsbar.

4. **Describe the code you wrote and talk about how you structured the code.**

   I min applikation så har jag klasserna, Avatar, Today, Dice, Dicegame21, Rock Paper Scissors, Menu och Main.

   Main-klassen är själva start klassen, den gör ett objekt av meny-klassen och kör dess metod .run()

Meny-klassen innehåller all logik för applikationen den har metoderna .printMenu(), .getChoice och .run(). run() metoden kör själva applikationen och använder printMenu() för att skriva ut en meny i användarens konsol, sedan använder den getChoice() metoden för att få användarens input (getChoice()) använder Scanner objektet. sedan har den ett switch case där den baserat på vad användaren skriver in 1,2,3A,3B,m eller q kör motsvarande kod.

Avatar-klassen hämtar en text-ritad figur från en fil som heter avatar.txt och printar den i konsolen.

Today-klassen använder java's inbyggda LocalDateTime och DateTimeFormatter klass och printar ut dagens datum.

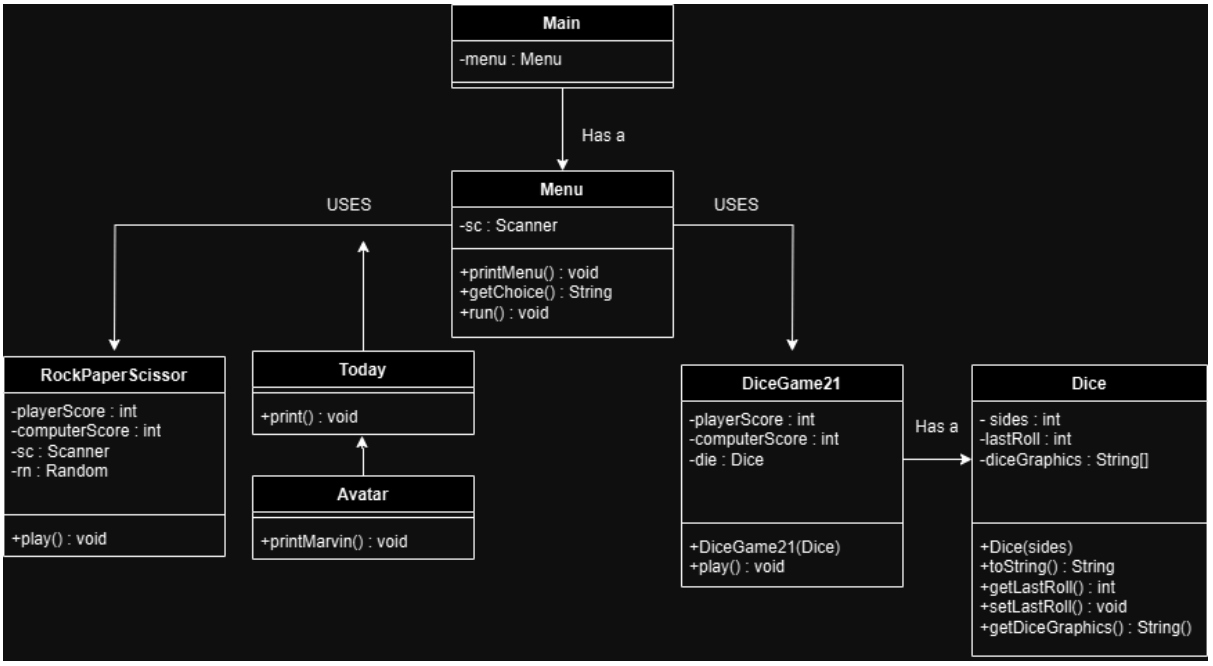
RockPaperScissor-klassen har metoden .run() som kör själva spelet. den instansierar en boolean variabel gameOn() för att hålla reda på när användaren inte vill spela längre. Variablerna playerScore och computerScore skapas också för att hålla reda på poängställningen. Spelet körs sedan med ett switch-case och använder Random klassen för att generera ett utfall till datorn, 1,2 eller 3 där 1 är sten, 2 är sax, 3 är påse. med hjälp av if-satser kan man avgöra vem som vinner varje match och därefter öka antingen spelarens eller datorns poäng.

Dice klassen har variablerna sides, lastRoll och diceGraphics. Den har även metoden roll() som med hjälp av random klassen stimulerar ett tärningskast och returnerar ett nummer mellan 1-(antal sidor) på tärningen. Dice klassen används sedan som ett argument i Dicegame21's konstruktor och skickas in med hjälp av dependency injection.

DiceGame21-klassen har också variablerana playerScore och computerScore för att hålla reda på vad man totalt har slagit med ens tärningar. Den använder också som RockPaperScissor-klassen en boolean variabel för att hålla reda på när användaren vill fortfarande spela. och Switch-case för att hålla reda på när användaren vill kasta en ny tärning eller väljer att stanna. När användaren väljer att stanna så använder jag en for loop som loopar 5 gånger och stimulerar 5 tärningskast för datorn (Marvin). Sedan med hjälp av if-satser så kan man kontrollera vem som har vunnit
(Den som är närmast 21)

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

Figure 1. Task_01 Class diagram



6. **What is your TIL?**

   I denna uppgift så har jag lärt mig hur man kan dela upp olika “ansvars områden” i olika klasser. Till Exempel så sköter RockPaperScissor klassen all logik för just det spelet, medans DiceGame21 sköter logiken för dice spelet. Menu-klassen sköter logiken för att printa menyn samt ta användarens input och Main-klassen syr ihop alla klasser med hjälp av ett switch-case

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?**

   Meny-klassen har hand om användare interaktionen och baserat på användarens val kör de de relevanta metoderna. Kod som används frekvent har jag skapat metoder till för att slippa skriva samma kod om och om igen.

   Till Exempel när man behöver se ifall ett team existerar så har jag metoden isTeamEmpty() som kollar ifall team == null. jag har också valt att bryta ner varje switch case kod till små metoder för att göra koden mer läsbar, Metoderna addSingleEmployee, addThreeNormalEmployees, addThreeSuperEmployeesWithPowers, and addSingleSuperEmployee har bryts ner till metoder som kan återanvändas, de tar ett objekt av Team som parameter vilket gör att man kan använda dem för att lägga till Employees i olika team. Att bryta ner logiken i metoder gör koden mer läsbar än om jag skulle ha all kod för logiken inne i mitt switch-case. Alla dessa metoder är även skyddade så de inte ska kunna användas i andra klasser och krascha applikationen.

   När användaren ska mata in en lön för en NormalEmployee så använder jag mig av try, catch med en while loop för att applikationen inte ska krascha ifall användaren matar in en sträng istället för en integer.

2. **Elaborate on how hard/easy it was to follow the class diagram.**

   Som jag skrev på task-01 så tycker jag att klass diagrammen hjälper en väldigt mycket för att få en “bas” av applikationen, sen får man själv implementera logiken.

3. **Do you understand the concept of "dependency injection" and did you use it in your application, if so, explain how you used it?**

   I denna task har jag inte använt mig av dependency injection, jag hade kunnat skicka in Team-klassen i Menu klassen konstruktor då hade jag använt mig av dependency injection, men i denna task ser jag ingen användning av det. Som jag har förstått de är det bra praxis att använda dependency injection när man vill testa sin kod, genom att använda dependency injection kan man göra “Mock” object (Objekt som man gör endast för att testa koden som de vore ett real scenario)

4. **Elaborate on what you find the most challenging parts with this task?**

   Det svåraste med denna uppgiften var att lösa logiken när man ska printa varje enskild i ett team i detta formatet -

   Emp(1): Little JR Doe (Trainee)

   Sup(2): Clark Kent (IT) - Flight, Strength

   Då det ska stå Sup ifall det är en SuperEmployee och Emp ifall det är en NormalEmployee, detta löste jag genom att overidea toString och work() metoderna i SuperEmployee-klassen och lät NormalEmployee klassen använda de befintliga metoderna i Employee klassen, så endast ifall det är en SuperEmployee kommer toString() och work() fungera lite annorlunda då de ska stå “Sup” framför och de kan ha superkrafter då måste vi loopa igenom superkrafterna och skriva ut dem också.

5. **What are your thoughts on writing comments like JavaDocs?**

   Jag Tror att det är en bra vana att skriva JavaDocs-kommentarer, särskilt när man jobbar i ett team. Det gör det enklare för andra att förstå vad koden gör, utan att behöva läsa allt i detalj. Kommentarerna hjälper alla att snabbt förstå hur saker och ting fungerar och vad varje del av koden är för. Det blir också lättare att hitta och fixa problem i koden när alla förstår vad som är gjort.

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

   Jag gjorde endast 2 av de frivilliga kraven, vilket var generate salary report och provide JavaDoc. Generate Salary löste jag genom att skapa en metod i team-klassen generateSalary() som tar dagens datum genom LocalDateTime klassen och sedan loopar genom varje anställd och formaterar en rapport med varje anställds lön och toString().

7. **What is your TIL?**

   I task_02 så har jag lärt mig hur man använder Inheritance och Composition. Jag har även lärt mig skriva JavaDocs kommentarer och generera en index.html med hjälp av kommandot “./gradlew javadoc” som blir en hemsida där dokumentation för applikation finns.

Task 03 - Interface, implements and file management

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?**

   Jag tycker att klassdiagrammet hjälper väldigt mycket, men det som hjälper mest är instruktionerna för varje requirement på uppgiften.

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

   Inheritance eller Arv är när två klasser eller flera har ett förhållande såsom att en klass “Är” en typ av en annan klass exempelvis -

   En bil ÄR ett fordon, en buss ÄR ett fordon, så i detta fall är “Fordon” en super-klass och bil och buss ärver egenskaper som gemensamt definieras för alla fordon.

   Composition funkar som så att klassen har instanser av andra “små” klasser som att där de har ett förhållande att den stora klassen HAR de mindre klasserna exempelvis

   Ett hus HAR fönster, dörrar, golv etc. så i detta fall hade man gjort en klass House och klasserna fönster, dörr, golv och i House klassen så hade man gjort objekt av de andra klasserna

   Ett interface för definierar metoder som vi vet att många klasser kommer att använda, det för definierar inte logiken för metoden utan bara själva metodens namn.

   En abstrakt-klass är en klass som man inte kan instansiera direkt, utan det fungerar mer som en mall och en gemensam bas för andra små klasser, där de tvingas att implementera samma logik. Genom att använda abstrakta-klasser så kan man säkerhetsställa att alla subklasser som ärver från den abstrakta klassen följer en viss struktur.

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

   Jag tycker att klass-diagrammet vi fick tillsammans med instruktionerna för uppgiften innehöll tillräckligt med information för att förstå och kunna lösa uppgiften.

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

Jag är nöjd med min kod, jag hade en del problem med att få det att fungera när “Vargen” skulle fånga spelaren. Men löste det tillslut. Jag valde att köra med den enkla strategin där vargen går slumpmässigt runt med hjälp av Random-klassen. man hade kunnat göra applikationen bättre genom att göra en strategi för vargen där den baserat på spelarens nuvarande position rörde sig mot hans håll istället.

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

Det svåraste delarna med att bygga denna applikationen var själva logiken hur spelare / vargen rör sig, hur man avgör ifall det är ett hinder i vägen för spelaren, hur man avgör att vargen tagit spelaren och hur man avgör att spelaren har vunnit genom att hinna fram till sitt hem.

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

Jag gjorde ingen av de frivilliga delarna i den här uppgiften, jag försökte göra en svårare strategi för vargen, där den skulle gå efter spelarens position men lyckades aldrig få till det.

7. What is your TIL?

I denna uppgiften har jag fått lära mig att jobba med abstract-klasser, interface. Jag har även lärt mig hur man serialiserade objekt, för att kunna “pausa” spelet stänga av det sen återuppta spelet där det avslutades vilket är häftigt.

Bonus points

Explain what optional parts you did in each task and summarize all the bonus points that you earned.
Följande är de frivilliga delarna jag har gjort :
task_01 : DiceGame21
task_02 : Generate salary report och JavaDoc
task_03 : Inga bonus poäng

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?

Jag gjorde endast första och andra labben, resterande koncept har jag lärt mig via youtube klipp, artiklar från exempelvis geeksforgeeks och guru99 och modulerna från canvas.

Jag tycker att det har varit jobbigt att gå från Python till Java alla små moment som att läsa från filer, skriva till filer är mycket mer komplicerat i Java. Eftersom vi aldrig fick testa objektorienterad programmering i Python så kan jag inte uttala mig om skillnaderna där. Men mycket som jag lärde mig i Python gick att föra över till Java, dvs hur loopar, conditions, funktioner osv fungerar.

TIL;

What was the most important TIL; (Today I Learned) you took with you from this assignment?

Det viktigaste jag har fått lära mig efter att ha gjort alla 3 tasks är främst objektorienterad programmering, Inheritance, Composition, Interface, Abstract osv.. Jag har även fått lära mig filhantering i Java vilket skiljer sig en del från python. Jag har också lärt mig hur man skriver kommentarer enligt java docs och att använda bygg verktyget “Gradle” vidare har jag även fått lära mig hur man både gör och hur man tolkar UML-klassdiagram.

References

[1] abstract class in java, Available:
https://www.geeksforgeeks.org/abstract-classes-in-java/

[2] inheritance in java, Available:
https://www.guru99.com/inheritance-in-java.html

[3] difference between abstract-class and interface, Available:
https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/