Reflection Note.

DIET MANAGER

We found the Diet Manager project pretty interesting due to the complexity of all of the elements to consider when forming our methods and the skeleton. Starting from scratch was new for us since we have mainly been working designing tests on methods and projects which were already formed for us. Luckily we had members in our group which have taken their time to do a lot of research prior to the project.

We were concerned before the project about a software which would be boring and not relatable for any of us. For example a software to manage the average size of all bird species. Luckily the software was understandable and we could picture most of the elements and functionality of the software at an early stage.

We also were fortunate enough to have a lecturer which answered all our concerns and questions when the exam was presented for us during the beginning of the project.

ASSUMPTIONS

We assume that a person can have 0 allergies. In this case allergies should be set to an empty ArrayList. We leave it to the customer to validate null inputs as the customer has not specified this in the requirements.d

Our focus is to achieve the minimum requirements. For that reason, we removed the getters and setters to achieve over 50% coverage.

At first we auto generated getters and setters in most classes, but we decided to remove the methods that we didn't use. We would include the getters and setters if we were going to keep developing the software, but in this case we chose to not.

When it comes to the range between two numbers in requirement 4b, we have interpreted it to include the endpoints.

NOTES REGARDING OUR SOFTWARE

- The test data in requirement 1,3 and 4 are initialized before each test case. The test data in requirement 2 is initialized in each test.

The benefit with initializing the data within each test case is that there's more control and also clearer to see with test data being used. The drawback for using this method is that there's a lot of repeated code. We had a discussion and wanted to show both methods. We agreed to show both methods and include it in our code.

Inputs that do not comply with the restrictions described in the requirements, such as attempts to input a non-vegan food into a vegan diet, the program will throw an exception error. Since we do not know what the customer wants to do in these cases, we leave the further handling of the exception error to the customer.

ADDITIONAL FEATURES

No additional features added to the software.

TEST TO INCLUDE FURTHER

Generated and not customized Getters/Setters tests
Generated and not customized Constructor tests
Null pointer test

THE DECISION-MAKING PROCESS

The first decision we had to take was how we were going to make sure that every team member would get the ability to code. We thought it would be important since every team member has different experience and wants every team member to both participate and also learn something. Some of our team members have limited java coding experience, so making sure that they had their own task was crucial.

All members also agreed early that we would treat the exam as a graded exam even though it is a passed/failed.

We landed on dividing coding tasks based on requirements. Every member would get the task that suits their experience and their personal preference. To make sure that every member has the same perception on what each task requires, we went through each requirement line by line.

Daniel was also responsible for which digital platforms to be used. We used Slack at the beginning, but realised that it would be too comprehensive and also excessive and would only reduce our productivity. We did never formally announce to stop using Slack, but it just became natural to not use it gradually over the project period. We mainly used Facebook Messenger for our non formal chatting and deciding meeting times and dates.

COMMUNICATION

The communication has mostly worked really well. Everyone in the group has been involved in decisions that were made. Not every decision applied to every team member, but every decision which was, was well communicated to every team member.

Some members couldn’t participate in every meeting or parts of the session and they would get a written report of every change that was made and also decisions about future meetings.

We have had a few communication issues regarding deciding if it was necessary with a physical meeting in favor of a digital meeting over discord. It was often because every team member was indifferent to which form of meeting they preferred. Our meeting responsible would therefore take the decision for each meeting based on our current status.

Every member has been available online and took part in the conversation. But differences in some team members' sleeping patterns has proven to be a challenge. We wouldn't always reach every team member either early in the morning or late at night. Our main and critical communication was therefore during the afternoon.

We also have set rules and guidelines for how the group work should be, but made sure to have an open discussion throughout the whole project period about any changes that were necessary to make.

Some of our members were coding in their spare time without informing the team which resulted in two members working on DietManagerTest at the same time. We solved this by picking out the best tests from both members. We all agreed to always inform the team if we were coding.

TEST PLAN

To exercise on writing a test plan we met right before the easter holidays to write a draft test-plan for the Event Narrator. We sat together to discuss each part and also make sure that we actually understood what each part of the test-plan requires. We gathered a handful of questions which weren't clear for Alberto on the Friday when the exam was presented to us.

We followed the scheduled time pretty well in the beginning, but after a few days it became more difficult to follow the time frames that we had set. We followed the scheduled tasks but had to postpone a few meetings but made sure to always be ahead of our sub-goals.

It was really helpful to give each member their responsibilities. We had to look back at the test-plan a few times during the project to double check who was responsible for what during our disagreements. The responsible would always have the last word.

WHAT IS NOT INCLUDED IN THE TEST PLAN

- Responsibilities for determining problem severity and correcting problems: This will be a part of "Responsibilities".
- Test schedules: Will be a part of the main "schedule".

PROJECT PROGRESS

Our project progress could be found in our change log and schedule in the test-plan. We feel like we would repeat ourselves by also including it in the reflection note.

DIVISION OF WORK

We mainly divided the tasks and responsibilities based on each member's own preferences. But we all agreed that one of the most important elements was for everyone to get the opportunity to code, with the help of others if needed. Some of the requirements were dependent on another requirement. Members would need to work together if that was the case.

Regarding more of the administrative tasks which weren't primarily a part of the project was optional. But some members had to take a task/responsibility since no one else wanted it. But we made sure that every member had some administrative work to do as well.

Regarding writing the common reflection note and the test-plan we had mainly 1-2 members which would be responsible for the writing itself with the input from other members when needed. That worked pretty well since some of our members prefer coding over text writing. That way we could both work on the text documents at the same time as other team members were coding. But all of the coders would still make sure to keep the text writers updated on changes to the code. All the written documents have been available on Google Docs for review if any members wish to do so.

As stated in our test plan each member had their own requirements either alone or with a partner.
Habil had the main responsibility for this requirement with the help of Daniel since he has a better fundamental understanding of how OOP works.
Harry and Elsa worked together on the second requirement since we thought it was quite comprehensive.
Daniel worked solo on the third requirement since he had a pretty good idea on how to solve it.
Isak had the main responsibility for the fourth requirement and needed some help so we all had to collaborate together to solve it after the first three requirements were completed.

Habil wrote the first draft and headings for the test-plan and gave each member 1-3 paragraphs which they had to fill out.

The reflection note was written by Habil with inputs and continuous reviews from other team members. We had a session where we went through what to include and what to remove afterwards.

OBSTACLES

One of the biggest challenges we faced in our project was to be on the same wavelength when discussing coding and implementation. Java is a pretty abstract language where each member has a different understanding behind the logic of the language and the system we're developing. As a result, we used a lot of unnecessary time discussing how to implement the tests and methods.

We overcame this by dedicating time to both explain and show our opinions during disagreements.

We made our own tests in our own branches in GitHub and faced some issues merging all of the branches. Daniel was responsible for all use of Git and solved these issues by himself since he's quite familiar with the use of Git.

PROBLEM SOLVING

We have had only a few technical issues mainly regarding the integration between GitHub and Intellij. Some of our team members have limited experience with using GitHub and Daniel was mainly responsible for helping members setting-up and also troubleshooting errors. The few times Daniel wasn't present we had to do our own research when we faced merging issues. It was actually a good thing since every member had to learn how to solve these issues without the help from Daniel.

Some of our members weren't able to design and test some of their methods. The easiest way to solve it was to have a joint meeting where everyone worked on the same method to try to solve it. Whoever solved it would share their screen and explain what changes were made to solve the issue.

Some members couldn't participate in all of the physical meetings due to personal errands. For example a team member had to travel back to his hometown for a week. These issues were mainly solved by having him on Discord while the other members were having a working session together. It worked better than expected and we realised that it wouldn't be an issue if a member had to participate from another location.

We had a handful of less important problems which were mainly solved by whoever was responsible for that issue. It worked pretty well and we were pretty happy to have defined our roles early in the project.

STRENGTH AND WEAKNESSES

Our group members were pretty different which we learned was a good thing. Since every member could focus on the tasks that they felt comfortable with and were best at. This way we always made sure to maximise productivity.

Each member's personal strengths and weaknesses were revealed during the project. The skill areas that the members will still need to develop is up to each member to reflect over on their personal reflection note which is included in separate documents.

WHAT WE ACHIEVED AND LEARNED

We are pretty happy with what we achieved. Every member has approved the code and hopefully learned something on the way. We gave ourselves a pretty tight and optimistic time frame for when we want to finish the whole project and we made it in time.

We were told earlier this semester that we had to Google a lot and we have had to Google a ton to finish the project. We underestimated this claim, but realised pretty quickly during the first week that it would be necessary to finish the project.

Also the schedule doesn't always go as planned and we learned that that is fine. We adapted quickly and worked overtime if necessary and changed deadlines if necessary.

WHAT WOULD WE CHANGE NEXT TIME

First of all we would go through naming standards for variables and methods. That would save us more time in the long run since we divided our tasks based on requirements without actually agreeing on naming. It took a lot of time and effort to rename and make code understandable for others than the coder himself/sheself.

We would also plan if a meeting was physical or digital the night before. Discussing it over Messenger the same day as the meeting takes place only leads to confusion and frustration.

Some of our members realised that Java requires much more time and training than first expected to be able to design methods and tests.