Tittel og undertittel med beskjæringsmerkegrafikk

Forfatter- og firmanavn med beskjæringsmerkegrafikk

**Table of contents**

[***1.***](#_gjdgxs)***What will be tested 2***

[***2.***](#_30j0zll)***Test phases 2***

[***2.1***](#_1fob9te)***Unit testing 2***

[***2.2***](#_3znysh7)***Regression testing 2***

[***2.3 Integration testing 3***](#_2et92p0)

[***2.4 System testing 3***](#_tyjcwt)

[***2.5 Approval testing 3***](#_3dy6vkm)

[***2.6 Error testing 3***](#_1t3h5sf)

[***3.***](#_4d34og8)***How to test 3***

[***4.***](#_2s8eyo1)***Plan for testing*** *5*

[***5.***](#_17dp8vu)***How to document test results*** *5*

***6. acceptance test 6***

# What will be tested

We are developing a system for managing a home exercise program, the program will have four different exercises: Strength, Endurance, Flexibility and Balance. Each exercise has a given value for intensity, sets, duration and equipment. Each person has intensity level and exercise they prefer. The following test will therefore consist of:

* Test if every exercise has values (intensity, duration, sets and equipment).
* Test that the user input in intensity level is an accepted value (min 1 max 10).
* Test that we can use decimals numbers in strength class, so the user can choose to use 17.5 kg If he wants too.
* Test that the user only can choose an intensity level that has a positive number.

# Test phases

Testing is a very important phase of the development process, errors and deficiencies will be detected at this stage. There are many different phases of testing that will be described here.

# Unit testing

Unit testing is the testing of each method and function individually. This type of testing will be used along the way of the code being created. These tests will run from the “Test” folder we created.

Our plan is to create unit tests for parts of our code to prevent bugs. For instance we would like to write tests to prevent intensity to be less than 1 or more than 10. This is to make it clear and simple when we are dealing with intensity. We would also like to implement tests on methods that are checking if the program is balanced, highest intensity, arrayList sorted by intensity, etc.

# Regression testing

Regression testing is used when we want to test that changes in the code doesn’t have any “devastating” effect on the code that already exists. These tests will run every time we add changes to the code, we cannot move forward before the code runs as planned. This is a crucial part for the compotation and development of our system.

# 2.3 Integration testing

Integration testing is about putting the different parts of our system together and testing them as groups. In our group we have divided different people into different sections of our system. It is therefore important that the compotation of the different parts is done correctly, so that we are left with one well-functioning system at the end.

# 2.4 System testing

System or validation testing is the phase which the entire system is tested against the requirements. This test is performed by the entire team. Its therefore important with good communication on what is to be tested, and that the code is commented on. The performance of the testing should be thoroughly documented, so the group easily can fix any problems that may occur.

# 2.5 Approval testing

In this phase a user or customer can test the system to determine if they are satisfied with the work that has been done according to the specified requirements. This type of testing is not relevant to our system.

# 2.6 Error testing

In this phase our system will be tested for all types of scenarios, our goal here is to “crack” the system. Inputs where the behavior of the system is not as expected, or when the system does not respond as expected, will be uncovered in this phase.

# How to test

To start with, we make manual test where we calculate values and make sure that they are correct. When testing our system further we well be using IntelliJ built in test function Junit 4.13, all our test will be made in an Exercise package, inside folder “Test”. Our goal is to have a coverage of over 50% and that all of our test passes.

# Plan for testing

The group will test the code on a regular basis, all member should be a part of this process. Before testing, expectations of how our system will handle the test will be discussed, and it should be clear what to test each time. Expectations should be documented and compared to the result.

# How to document test results

Documentation is an important part of testing, we should always in advanced write a test case. This specifies where in the system we are testing, and what is being tested, here we can also document our expectations and compare it to the result. We are using Google Docs to document the testing, in Google docs members can document test cases with screenshot of the test code and result. other members of the group can always see the newest update of what has been done. We will also create a excel sheet that document the different test cases (name, date, testcase name, and if it passed or not).

**Acceptance test**

|  |  |  |
| --- | --- | --- |
| Requirement | Criterion | Test Cases |
| 1. | a. Each program keeps an account over the highest intensity of the exercises in the program.  Also prevented intensity to be between 1 and 10. | highest Intensity()  negativeNumbers()  intIsEmpty()  negativeOrEmptyInts() |
|  | b. Checks if program is balanced: Program is balanced if it contains one of each type. | exerciseNotBalanced()  exerciseIsBalanced() |
|  | c. Checks that arraylist program is sorted increasingly by intensity. | sortByIntensity() |
| 2. | Checks that each person is given a suggested exercises for the level of intensity (+/- 1) and exercise they prefer | exerciseIntensityIsCorrect()  testThatProgramIsAppropriate()  checksScoreFromPreferredExercise() |