Creation of the Potato Language

Mark van Doesem & Tim Sonderen

S1496468 & s1465252



# Main Features

## Story Telling

Potato code is designed to look a lot like normal text, so you can literally tell the program what to do in relatively normal English. It works with capital letters for names and dots at the end of each line to simulate the feeling of writing normal text instead of code even more. For example a declaration is constructed with ‘suppose’ and a variable name and a task (function) is declared as ‘Task’ , then the variables it takes in correct comma and ‘and’ usage and then what it gives back.

## Comments

Of course, you can also create comments in Potato code. This comments are completely ignored by the compiler, but can be useful to explain what you code is supposed to do at a certain point. A comment is started with the keyword ‘btw’ and is ended with a dot. Quite a nice feature about these comments are that they can be put anywhere in the code. So for example you can put a comment after each argument of a task that explains what that argument should represent.

## Tasks

Potato works with tasks so that you can work more efficiently because you do not have to duplicate any code if you want to reuse it. This way less lines are required for a more majestic piece of code.

## Arrays

In Potato, you can use arrays to store a list of values so you do not need to a separate variable for every value. You can use the elements of an array just like any other variable. This way you can for example loop over an array to check if each value in that array is greater than a certain number, which could be useful for checking test results.

## Increment

Potato also supports task increment. This task, of course, increments the given variable. Increment takes only a variable because incrementing an integer wouldn’t be saved anywhere. The Increment task can be very useful in a ‘while’ loop, for example to iterate over a list or to count something.

## Errors

The Potato compiler is able to give you several errors at once when you accidentally made a mistake in your code. It will try to compile as much as it can and then print a list of errors if there are any. This list is clearly ordered with the sentence number in front of each error so it is clear where and what went wrong.