# Assignment 1

## Introduction

**Overview**
We are doing the jCalculator project track. This system is a plugin-based calculator which performs all calculations via plugins. The main type of user for this calculator is any person (human) who wishes to perform a calculation. It is recommended that the user collaborate with a programmer (or be one themselves) so they can develop their own plugins. Our calculator will work on any number-based calculation, provided the appropriate plugin is provided.

**What does it do?**
1. The user begins the program and is given a prompt containing all of the available commands
2. The user may begin inputting any command they wish, one at a time
3. When the user inputs a command, the result is printed back to the command line and the user may go back to step 2

**How will it work?**
We will use a command-line interface for the user to interact with. The main calculator module will contain code responsible for the following; prompting the user for a command, reading in the command, and executing the command. All of the commands except the ones stated here will be found in plugins, each with their own name and contract that gets added to the system when a plugin is enabled.

**What modules will support this?**
1. **Calculator:** This module shall be the main module responsible for running the calculator program.
2. **I/O:** This module shall be responsible for interacting with the user by printing output and receiving input into the program.
3. **Plugin Port:** This module will contain all of the basic commands, as well as add to its list of commands by connecting the plugins.

## Features

### Functional features

The functional features include the foundational components of this program. When deciding which features to include, it was important to consider what would specifically be needed for the

program to be implemented as designed. The user commands are a core feature of the program given that they are the means through which the user can interact with and operate the calculator. Next, we had to be sure to specify the requirements for the program to be able to read what the user had inputted. It is also crucial for the program to be able to parse through the necessary executions and write its output. The redo and undo feature requirements will grant the user the flexibility needed to use the program confidently. Finally, since the program is largely structured around various plugins, it is important for the users to be able to understand and parse through which they have installed.

| **ID** | **Short name** | **Description** |
|---|---|---|
| F1 | Commands | The player can perform calculations by interacting through the command line. The available command-names shall be the following:<br>- addPlugin [plugin]: adds a new plugin, which incorporates all of that plugins commands<br>- help: provides a description of the program, its purpose, and a list of commands<br>- plugins: provides a list of available plugins (maybe also an indication of which have been added by the user) |
| F2 | Read Operations | reads all commands from the plug in and displays possible operations. |
| F3 | Execute Operations | given an operation name and an appropriate number of parameters, executes the operation from the plug in |
| F4 | Write Operation | The result of the executed operation is displayed in the command line if a success or an error statement if the operation fails to execute. |
| F5 | Undo | Assuming that the latest function called was not the first operation, recalls the previous operation and displays the operation and result in the command line. Otherwise displays an error. |
| F6 | Redo | Assuming that the undo command has been successfully called at least once, recalls the operation that had followed the currently displayed operation and displays the operation and result in the command line. |
| F7 | Print | User should be able to print the list of imported plugins |

| | | |
|---|---|---|
| F8 | Plugin manager | User can import,create,remove plugins from the CLI |

### Quality requirements

When determining which quality requirements would be most significant for this product, we had to consider the user's point-of-view and ensure that their experience with the program is frustration-free. The ability to input commands is the foundation of the entire program, so we decided that the reliability sanity checks are the most pressing quality assurances to addressed. By committing to an intuitive UI and a stable program as core quality requirements, we will ensure that the program is built with the user in mind from the very beginning of development.

| **ID** | **Short name** | **Quality attribute** | **Description** |
|---|---|---|---|
| QR1 | Commands sanity checks | Reliability | When the player issues a command, the syntax of the command shall be validated against the format specified in F2. |
| QR2 | Easy to operate | Usability | The program's command-line interface should provide the user with an intuitive experience with the calculator |
| QR3 | Persistent plugin storage | Maintainability/Reliability | Imported plugins shall remain stored in the program's directory until the conclusion of the program |

## Java libraries

The libraries we planned to use did not end up being efficient, so we have omitted this section.