**Shticell: Exercise 1**

**Submitted by:**

**Liri Jacob ID:208606558**

**Tamir Reuven ID:208134544**

**Relevant email address:** [tamirre@mta.ac.il](mailto:tamirre@mta.ac.il)

System loading and saving bonus implemented.

**General description:**

We decided to divide our project into 3 different modules.

1.System engine module: this module is where all the logic and the bare bones of the system are implemented: the different functions you can use with the sheet, the cell and sheet interfaces and implementation and more that well cover later.

2.UI module: this module oversees the flow of the system from the user perspective, where the menu is configured and where the inputs are taken.

3. DTO module: when we finished the System Engine module we designed, we realized that the UI and the Engine need a way that they can communicate with each other, and there came to our aid the DTO module! Each command that the user wants to perform, the UI uses the engine logic to perform it, and gets in return a DTO object containing relevant information about the command that was performed.

Now well describe vital parts of the System Engine and UI modules and how we designed them.

**System Engine module:**

1.Sheet and Cell classes: we decided to implement the cell in the form of an interface and a class to implement it when each cell holds a cell identifier: a class we created to hold the row and column of the cell). About the spreadsheet, we decided to limit the access of the sheet update abilities in some of the functions. So, we created a structure: a read only sheet interface, an update sheet interface and a general sheet interface that extends both. And like that we can control how much we want to expose our sheet to the different functions.

2. functions design: we decided to implement the different functions (plus, sub, ref etc.) in this way: we created an interface called expression, and three interfaces that implements it that characterize all our functions in terms of number of arguments: Unary, binary and ternary expressions. And then, all the different functions implemented their appropriate interface (for example, plus implemented binary expression, abs implemented unary expression etc.). In order to recognize which, function the user wanted to use, we created a class called Function parser. Which helps us determine that.

3. main commands implementation: we decided to create an engine interface with the different commands we want our engine to perform such as: display a cell or display a sheet. And implemented that interface, and like that we didn’t couple the UI to our specific engine implementation, and we can use a different implementation of the interface with minimal code change.

**Important notes about the module:**

* After a discussion in the forum, we implemented that any use of function with an empty cell will result in a Nan or Undefined values (depends on the function).
* After a discussion in class, we decided not to fail the whole move if a function received wrong types of arguments, the result will be Nan or Undefined, again, depending on the function, also in the loading of a file we didn’t fail the load for using wrong types in function arguments but you will see Nan or Undefined. We did fail the whole move if the number of arguments was not as expected or trying to use a function that does not exist.

**UI Engine module:**

1. menu design: when we decided to implement the menu, we decided to not be coupled with one menu, so we decided to create a menu interface, with a class to implement it.

2.Inputs and communication with System Engine: as we described before, we used DTOs to communicate between the two modules, and the System Engine will not print information or take input from the user, so we decided to implement a version of implementation of the possible commands (loading a file or choosing to display a specific version of the spreadsheet). An implementation where we will take the input from the user, present relevant information and errors. And within that implementation, well call the engine implementation of the command logic part and use the DTO received from that to gather information about the command that was preformed (succeeded or failed, or a specific cell details) to display the user relevant information.

In conclusion, we had fun starting with the project, we felt we gained a better understanding of the process and the design part of a project and features, and a better understanding of OOP in general. we are really looking forward to continuing the process.