Bitcode Assignment 3

1 Design Patterns

Using design patterns in software project is a good practice. It helps to make your software understandable, sustainable and expendable. We have chosen two design patterns and implemented them in our existing code, the state pattern and the strategy pattern.

1.1 The State Design Pattern

why? tbd

1.1.1 State Class Diagram

tbd

1.1.2 State Sequence Diagram

tbd

1.2 The Strategy Design Pattern

why? tbd

1.2.1 Strategy Class Diagram

tbd

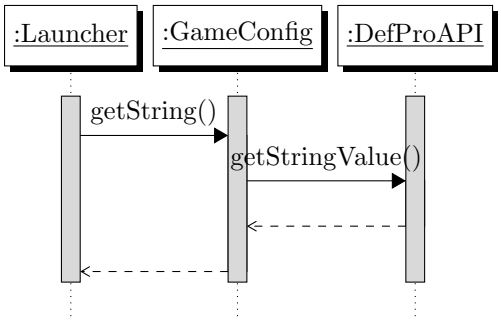
1.2.2 Strategy Sequence Diagram

tbd

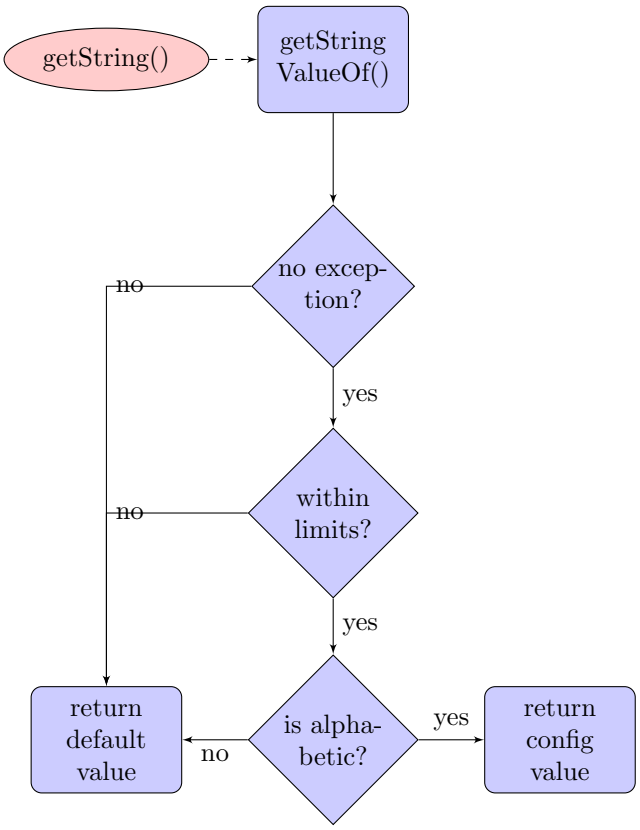
2 Defensive Programming

In the previous assignments it was required to use a game configuration file for initializing variables using a provided library. However, the provided library introduced bugs in the game.

To solve the bugs we have created a wrapper class (CameConfig) around the library's API that catches all the bugs and checks if the variables are within limits. This means that for every different API call we created a method in the GameConfig class. In the sequence diagram below is shown how the wrapper class works.



In every method in the CameConfig class that implements the API checks are build in to verify the data that is returned by the API. Also if the API throws an exception it is catched in the method. If an exception occurs or the data returned by the API is not within the defined boundaries the method will return the defined default value. The flowchart below shows how the method getString() is implemented.



In this particular example the following conditions makes the method return the defined default value.

- If the library thorows an exception.
- If the amount of characters in the sting is less or more then expected.
- If the string is not alphabetic.

3 Selectable Difficulty

One way of making the game more competitive is to implement selectable difficulty. The idea is that the user can select a desired difficultly between easy, normal and hard before the game starts. The selected level of difficulty will noticeably influence the difficulty of the game.

3.1 Requirements

The requirements are ordered by the MoSCoW model.

Must Have:

- The game must have three levels of difficulty, easy, normal and hard.
- The player must be able to select the desired difficulty befor the game starts.
- The game must have a start screen.

Should Have:

- The player should get more points when removing a tile or a background tile when the game is more difficult.

Could Have:

- Higher difficulty could mean that the player should find four of the same tiles on a row.
- Higher difficulty could mean that the player has less moves to remove all background tiles.
- Higher difficulty could mean that there are more background tiles.
- Lower difficulty could mean six different tiles instead of 7 tiles.

3.2 Software Design

The figure below shows the class diagram of the implementation.