FINAL REPORT

1. Problems Encountered

a. Working as a Team

Working as a group sometimes got really challenging. She has proven herself to be a great team member with full cooperation. However, we had a little argument about the implementation between some of our group members resulting in lack of communication and unbalanced workload. Especially, sticking to the reports got harder, for some members who helped during the report phases were no longer communicating with the implementing members. Because of this we had to design the project almost from scratch.

b. Lack of Usable Design

Starting from scratch led to many design related issues. We had to keep in mind the number of members that are implementing and shrink down some of the classes, thus, the design we have created for our project was no longer usable. We had to redesign most of the project, trying to be as efficient as possible, for we had really limited time and lesser members than expected.

c. Implementing in Java

Not being that familiar to implementing a full game, it was challenging to keep things in control. One of the main problems was that neither of us really remembered(/knew) implementing GUI in Java.

2. Solutions To Problems

a. Working as a Team

We ended up working on our project as 3 members rather than 5, so we had to make some trade offs. Especially, sticking to the reports got harder as explained. So, we decided to make some major changes about the design of our project. The three of us tried to balance the workload as much as possible, while also keeping in mind some parts of implementation was not suitable for division among individuals.

b. Lack of Usable Design

The previous problem made our initial design ineligible. Therefore, we created a new design with the present members, which was really hard for we had very little time. What we have done is that we divided the parts very carefully, allowing individual work and design. But, we also made sure that every part that is created were compatible after every step.

c. Implementing in Java

Our solution was once again careful division of the workload. Tibet was responsible for GUI, while two Irmaks worked on the game logic. Thanks to this we limited the part that we had to revise/learn about Java.

What could have done in other way was to design our classes together at the first place. Our some members who mostly contributed to the analysis report's object and class diagrams in their way which forced everyone to design it in that way in design sate. However, it's possible to see much more easier ways when the implementation is in our hands. Since it was possible to implement it in much more easier way, the project's implementation does not cover exactly same as the designs and diagrams which also caused another trouble that we were left without design which enabled us communicate in the first place. The contribution and the communication of every each member in every each state should be a must in designing engineering projects. We lacked it, which caused the trouble of designing project in such a way that only one person could work on it however, since there was only a few working on the implementation, it did not cause such a huge problem. Only that, the person who wrote the logic had tons of workload.

3. The Changes Made

There are countless changes that are made like classes, methods and design.

Game Play / Logic:

The game logic was implemented different than we designed. First of all, we used boolean array to determine our whole game since it would be much more efficient than implementing our program with integers. Moreover, we changed the objects:

There are 3 types of objects:

Block: which is the current falling object and it consists of either one of the shapes (I, J, L, O, T, S and Z)

Pit: This holds the value for stable unit squares which are bottomly spaced on the gameplay screen. The pit stays consistent until next block has been added to the stable units. (When current falling object collapses with a bottom object of the pit)

Graph: Graph is what we see on the screen at the moment. It changes rapidly with the timer and gathers information from both of the block and the pit.

The functions between these 3 classes form the game logic. What is given to the interface panel is the graph's current status. Also, notice that Timer of javax.swing been used in implementation. The timer action listener is called in the game logic to determine fall whereas inputs are gathered from the panel's action listener which is a keylistener. The reason we done this was to adjust the game considering user input, logic and the output. The output is given from the panel, keyboard listener also gathered from the panel whereas timer needed for logic so it placed under the game logic class. Since all of the logic was implemented by one person , there was not a need of much communication between classes. However, it was important to give logical names to variables when there was a need of help of another member. Tibet and Irmak A. helped debugging the code when needed. However, since there was lack of comments at the first place, explanation needed when debugging with the other teammates. It is really important to handle the problems / making debug together since someone can notice others mistakes or even making explanation to a person may help the code's owner understand where the mistake was.

Main Menu(GUI):

4. How Implementation Went

Implementation got devastating sometimes, but we did not give nonetheless. Especially, the parts that we were not that familiar like GUI gave us a hard time but we handled it. Our interrupted communication and understanding of each other helped us a lot, the three of us succeeded in working as a group.

5. Final State of Implementation

We have achieved most of the things we have planned. Our opening page is working great with all the buttons except the help option. We did not have enough time create a tutorial, so that button does not lead to anywhere. When the new game button is started it successfully opens up a new game. But the modes are unfortunately not implemented.

6. User's Guide

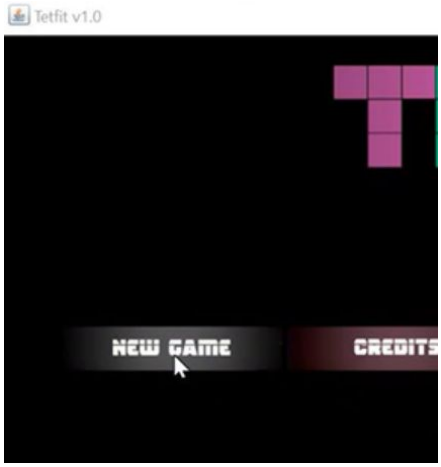
7.

Installation

The game is a simple java (jar) executable file which has no need installation. It is possible to move the file anywhere and execute.

Game

When this file is executed the player will encounter a main menu before the gameplay for some options to change the settings or check the high scores.



Once the player press the "New Game" button the game screen pops up.

Controls

The player controls the blocks with arrow buttons:

UP : Turns the block to left

DOWN: Turns the block to right

LEFT: Moves the block to left

RIGHT: Moves the block the right