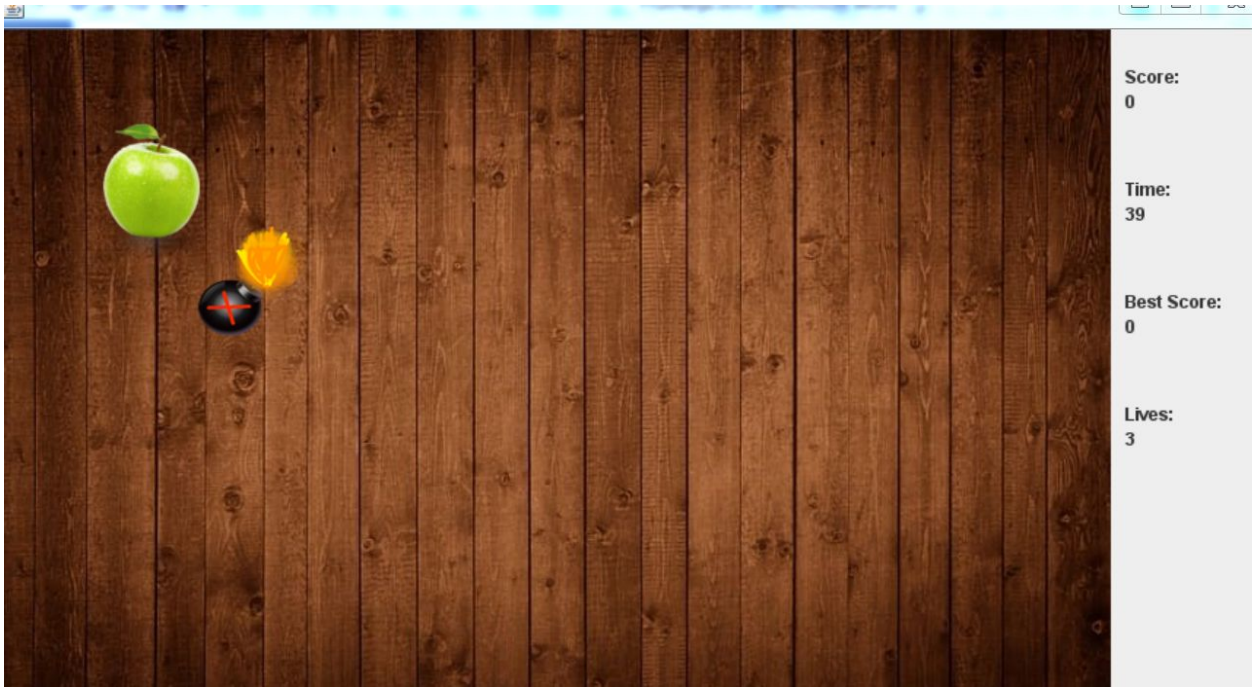
Report:Fruit Ninja

1-Through description of the design:

When you first start the game you have to enter your name and the level of the game (easy, normal or hard).

If you’ve played the game before your last best score will appear to you and you’ll win if you beat this score.

If not you’ll start with an initial best score of zero.



We have the fruits (apple, green apple, lemon, strawberry, watermelon and orange) as our first type of objects, which are sliced by the user. The second type of objects are the bombs,

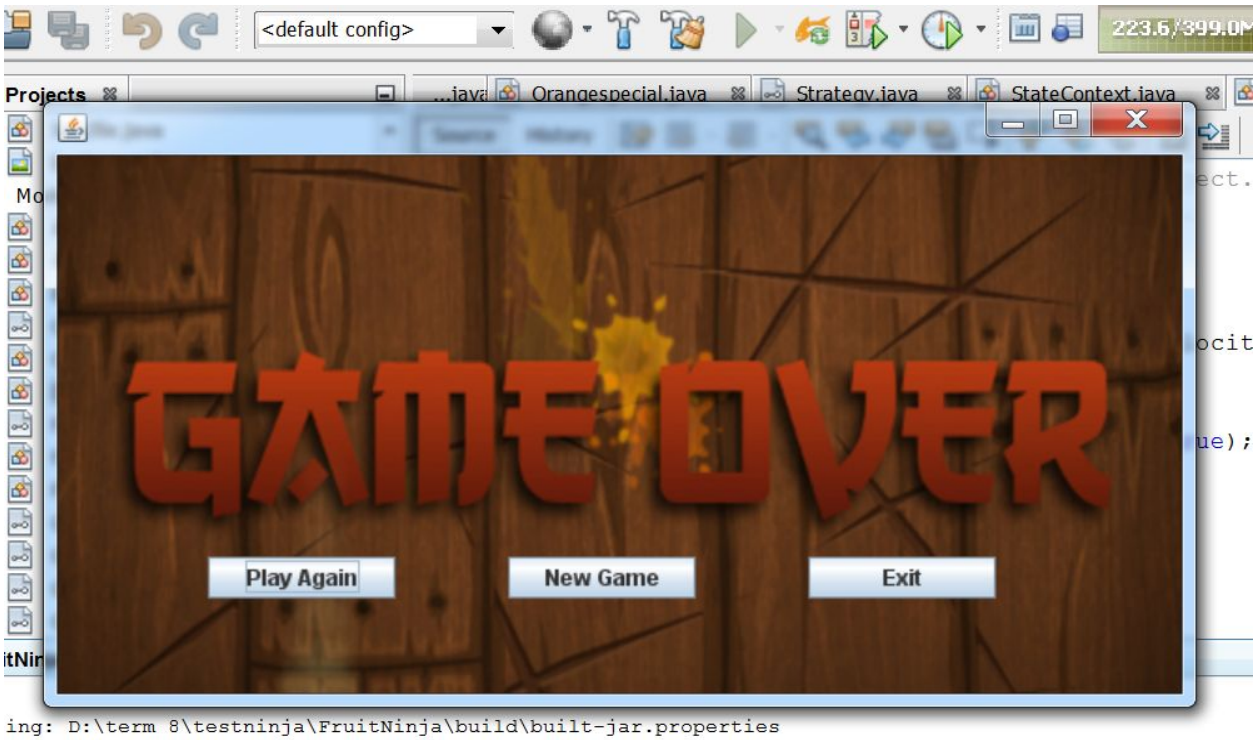
we have the dangerous bomb and the fatal bomb. All methods applied on both types of objects are created in the **GameObject Interface.**

In interface **Factory** we have the two methods **CreateFruits** method and **CreateBombs** method which create these two types of objects according to the given name of the object as input in the method.

Each of the above mentioned objects has a class in which the movements of the object, its initial velocity, its final velocity, when it's off the screen, when it's inside the screen, when the fruits or the bombs get sliced what happens, are controlled.

In **class Mouse** controlling the slicing of the objects whether the bombs or the fruits takes place. When the user clicks or swipes on the objects two things happen, first one is that the object gets sliced if it's a fruit and the score increases by one point if it's a normal fruit or by five points if it's a special fruit (watermelon or orange). However, if the user slices a dangerous bomb or when he drops a fruit without slicing it he loses one life, if he lost all three lives he loses the game or when he slices a fatal bomb he also loses the game.

If this happens and the user loses, a new window appears where he gets to choose to play again or exit or play a new game which directs him to the first window at which he enters his name and chooses the level of the game.



Each Player who enters the game his name, his score, the chosen level and the number of lives get saved in the xml file, that's why if a player who played before enters the game his last best score appears again.

```
<?xml version="1.0" encoding="UTF-8"?>
<players>
  <player>
    <name>mariam</name>
    <score>91</score>
    <lives>2</lives>
    <level>0</level>
  </player>
  <player>
    <name>plank</name>
    <score>91</score>
    <lives>2</lives>
    <level>1</level>
  </player>
  <player>
    <name>theBestGamerEverrrr</name>
    <score>27</score>
    <lives>1</lives>
    <level>2</level>
  </player>
  <player>
    <name>menna</name>
    <score>48</score>
    <lives>3</lives>
    <level>0</level>
  </player>
  <player>
    <name>m</name>
    <score>0</score>
    <lives>3</lives>
    <level>0</level>
  </player>
</players>
```

2-Design Patterns:

In our game we used the following patterns: Singleton, Factory, Command, Observer and Strategy.

1-Singleton:

We applied the Singleton design pattern by having a class named **Player** because each time only one player plays at a time so we have only one instance from this class in the whole game, which is the intent of the pattern.

2-Factory:

In this game we have two types of objects the fruits and the bombs so we made one factory that is responsible for creating any type of objects. So, we refer to the newly created object through a common interface (Factory) but we let the subclasses decide which class to instantiate whether **FruitsFactory** or **BombsFactory**.

3-Command:

In order to apply the Command design pattern in our game we used the interface Command which includes the method **execute.** We also have two classes **MovedownCommand** and **MoveupCommand** that implement this **interface.** These classes invoke the operation without knowing how to perform the operation move up or down, which is the intent of the pattern.

How to perform these operations is only known by the object itself (the fruits and the bombs). This way we separated the object that invokes the operation which is in classes **MovedownComman**d and **MoveupCommand** from the one that knows how to perform it which are functions **Moveup** and **Movedown** that are implemented in each class of each object.

4-Observer:

We applied this pattern by having the **interface Observer** which contains the method **update** which is implemented in class **GameFrame.**

When one object which is the Player changes its state, all its dependents which is in our case the GUI, class **(GameFrame**) is notified and updated automatically.

In class **GameFrame** we have the function update which updates any changes that happen by the player like changing the score, the time and the number of the lives. When these change their labels are automatically also changed as all observers are notified with these changes but in our case we have only one observer .

5-State:

We used this pattern by Defining interface **State** for declaring what each concrete state should do.

We provided 2 concrete states implementing the State interface:

1-**Winning:** Initial State.

2-**Losing:** when we have zero lives or sliced a Fatal Bomb.

StateContext class Defines an interface to client to interact. It maintains references to concrete state object which will be used to define current state of object.

And also define the state of the level chosen, zero for the easy level, 1 for normal and 2 for the hard level.

6-Strategy:

We used this pattern by having an interface Strategy that includes the function slice which slices the objects and modifies the state according to that.

If the Object type is a fruit then the slice operation inside the **SliceFruit** class will be called. And so on for the fatal and dangerous bombs.

3-User Guide:

How to Play:

.·Enter your name and choose the level of difficulty.

·Swipe your mouse across the screen to deliciously slash and splatter fruit like a true ninja warrior

.·Be careful of bombs, they are explosive to the touch and will drop your precious life.

Rules:

.·Start the game with 3 lives.

.·.Ones your lives reach zero you will die.

.·Beat your highest score to win.

.·Points are obtained by slicing the fruits.

.·Your life will be damaged if you missed a fruit.

.·Your life will be damaged if you hit a dangerous bomb.

·You will die if you hit a fatal bomb.

Objects Description :

.·There are two types of bombs:

1-Dangerous:

- Dangerous bombs will damage you by decreasing your lives.

2-Fatal:

- Fatal bombs will immediately kill you.

.·There are two types of fruits:

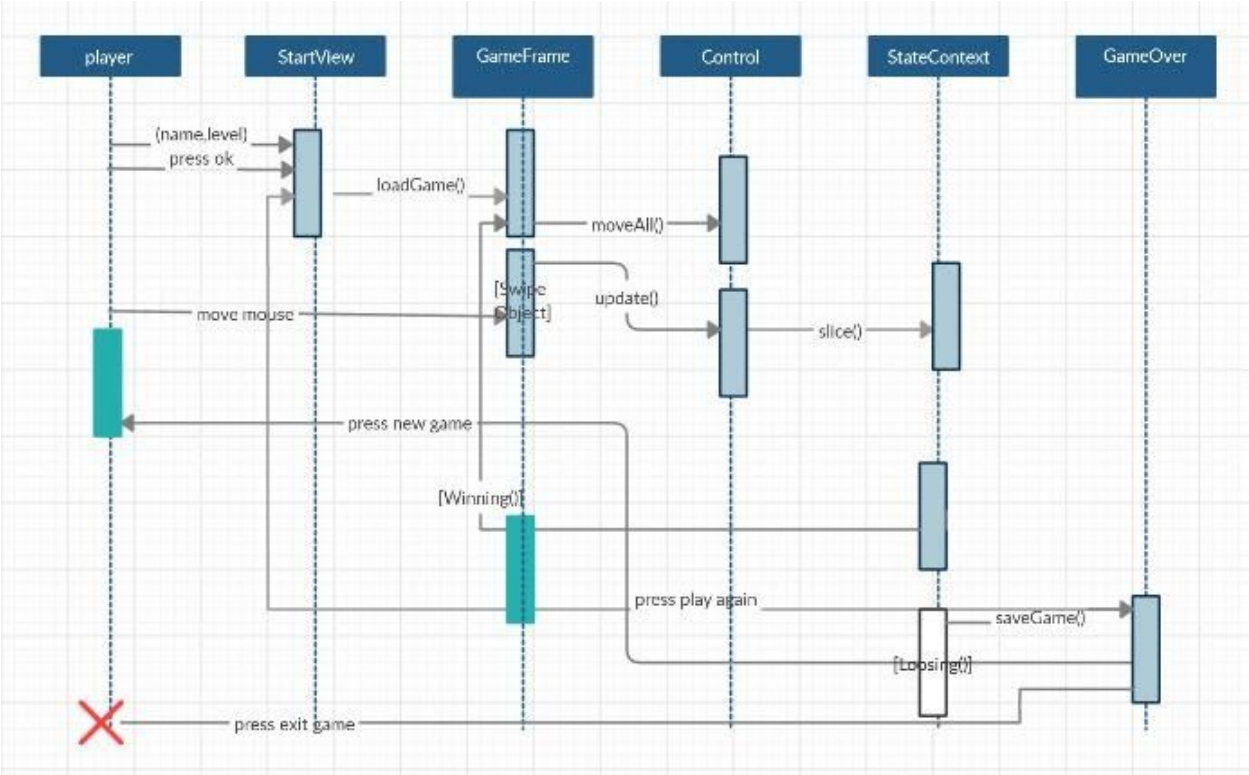
1- Normal:

- Normal Fruits give you 1 point each.

2- Special:

- Special Fruits give you 5 points each.

4-Sequence Diagram:



5-Class Diagram:

check the Class Diagram pdf.