Fruit Ninja - Game

Objectives

We have studied many design patterns and it is time to apply. As Design patterns is one of the topics you will always be asked in interviews and you will always need during real life coding, we highly encourage you to invest some time in this project.

Remember, a good, maintainable and well structured code will never be free of design patterns, So a good programmer should have a good understanding of Design Patterns.
So try to practice applying design patterns in this project, also we would advise you to put this code on a public github / gitlab repository so it will add to your CV later after graduation.

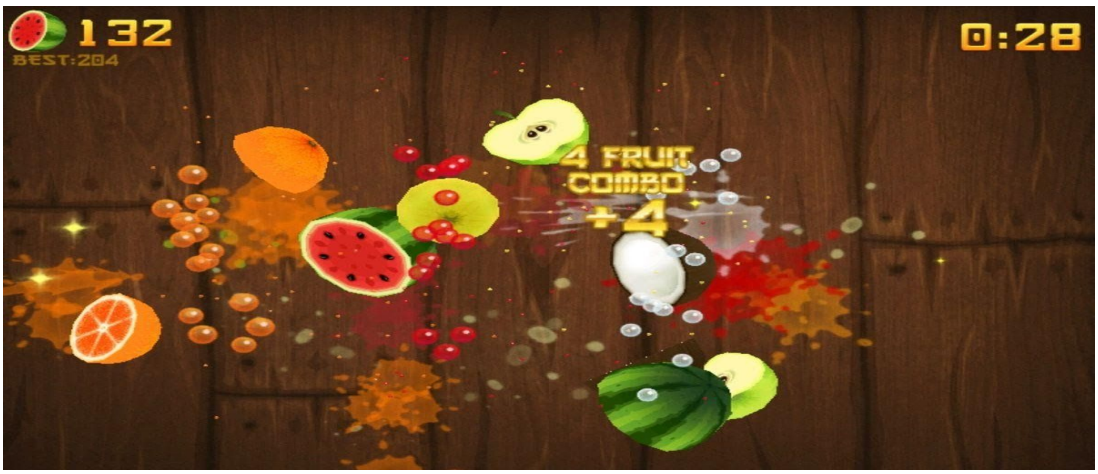
We really encourage you to practice applying design patterns in this project, this project marks will be given as a bonus either on coursework or final exam and they will be upto 10 marks and may be more !

It is the final project for this course, let us put all that we have learned together.
We are going to implement a fruit ninja game. In this project you are required to:

- Design an object-oriented model for a full game.
- Apply the OOP concepts of inheritance and polymorphism to your design.
- Apply different Design Patterns to your model.
- Get to know the concepts of game programming.

Game Description

It is a single player game in which the player slices fruits with a blade controlled by mouse swipes (finger swipes on touch screen in case of smartphones). As the fruits are thrown onto the screen, the player swipes the mouse across the screen to create a slicing motion, attempting to slice the fruit into two halves.



Rules and Tasks

- You have at least the following:
  - 3 types of **Fruits**.
  - 2 types of **Bombs:**
    - **Fatal**: ends the game immediately.
    - **Dangerous**: makes you lose one life.
  - 2 types of **Special Fruits** that add special extra credit points upon slicing.
- Your goal is to beat the highest score, we will implement the classic version of the game. *i.e when you play for the first time you have an initial score of zero, as you play you try to beat this score,*
  `best_score = max(best_score, current_game_score)`

- You need to support 3 levels of difficulty, but you are free to choose any criteria for difficulty (different fruit speed, increase number of bombs, increase number of fruits, ... etc.).
- You should have a label to show your score. And this label should be updated so each time you slice a fruit, the score increases.
- You should also have a time label to show how long you have been playing the game.
- You should also show a label with the highest score you are trying to beat.
- You should also show your remaining lives: you have 3 lives in the game, you lose one life in one of those two cases:
    - The fruit drops off the screen without slicing it.
    - Slicing **Dangerous** bomb.
- You lose the game in one of those cases:
    - Slicing **Fatal** bomb.
    - Losing all your lives (happens when dropping a fruit or slicing a dangerous bomb).
- You play until you lose all your lives, so you win the game if you beat your last saved high score, initially your score is **zero**.
- You should also support saving and loading the game on specific state (score, remaining lives and fruits shown on the screen, level, etc..) using **XML** files.

- You should use at least **5** of the following design patterns:
    - **Singleton**
    - **Factory**
    - **Command**
    - **Memento**
    - **Observer**
    - **State**
    - **Strategy**
    - **Decorator**
    - **Adapter**
- You **must** use MVC**.**

Bonus

You can implement any of the following as a bonus:

- Playing audio when the game starts and playing slicing audio when fruit is sliced.
- Supporting **Combo swipe:** extra points are awarded for slicing multiple fruits with one swipe.
- Supporting **Arcade mode**: The player has only 60 seconds to achieve a high score, lives don't matter just slice till time is out.
- Supporting multiple players with players' score boards.
- Adding special fruits ex(fruits that can make the game slower for a certain amount of time, fruits that auto slice all fruits in the screen, ... etc)
- Any exceptional design or outstanding GUI.

Notes

- For simplicity, it is not required to throw fruits with different angles. you can assume that the X axis is always constant.
- This assignment mainly tackles the design issues, so take a sufficient time for designing the structure of your program.
- Delivering a working game is important but it means nothing if you don't use OOP and design patterns paradigms.
- A large portion of your grade will be on you design, make sure you follow **SOLID** principles, also make sure you separate UI from your program logic.
- Start in this assignment early and feel free to ask any questions.
- If you don't know the game you can find it on Google Play or App Store.
- You must implement at least the methods in the suggested interface below, however, you can add new methods, if needed.

Deliverables

- Work on this project in groups of 3 or 4 .
- Implement the assignment in Java.
- Delivering you code through GIT is **mandatory**, showing meaningful GIT log is important.
- Your submission should include:
    - The code folder
    - **Self-executable jar file (JRE 1.8)**: The program should be executable by simply double clicking the icon or by double clicking on a bash/ batch file .
    - **Report** should include a thorough description of your design.
    - **Report** should include a **Class Diagram** showing main classes in your application.
    - **Report** should include **Sequence Diagrams** showing essential or main behaviour among your classes, showing only necessary and important program flows.
    - **Report** should include the **Design Patterns** you used and the intent it is used for in your application.
    - **Report** should include snapshots of your GUI.
    - **Report** should include a user guide that explains how to use your application.
    - Add Readme.txt file with any assumptions you need to run the program, and with the detailed division of labor among the team members.
- Delivering a copy will be **severely penalized** for both parties, so delivering nothing is so much better than delivering a copy.
  Late submission is accepted for only 3 days after the deadline and will be graded from 80% of the full grade.

References:

- Falling Objects in JavaFX, sample code is in the video description.
- JavaFX Animation Tutorial
- JavaFX Animations TutorialsPoint
- Mouse Event Handling
- MVC briefly
- MVC with JavaFX Sample Code

Suggested Interfaces

These Interfaces are to guide you around, they are **not mandatory** to use.

```
public interface GameObject{
      /**
       *@return the type of game object
      */
      public ENUM getObjectType();
      /*
       *@return X location of game object
      */
      public int getXlocation();
      /*
       *@return Y location of game object
      */
      public int getYlocation();
      /*
       *@return max Y location that the object can reach on the screen
      */
      public int getMaxHeight();
      /*
       *@return velocity at which game object is thrown
      */
      public int getInitialVelocity();

      /*
```

```
     *@return failing velocity of game object
    */
    public int getFallingVelocity();

    /*
     *@return whether the object is sliced or not
    */
    public Boolean isSliced();
    /*
     *@return whether the object is dropped off the screen or not
    */
    public Boolean hasMovedOffScreen();
    /*
     *it is used to slice the object
    */
    public void slice();

    /*
     *it is used to move the object on the screen
     @param deltaTime: time elapsed since the object is thrown
                       it is used calculate the new position of
                       fruit object.
    */
    public void move(double time);
    /*
     *@return at least two images of the object, one when it is
     sliced and one when it is not.
    */
    public BufferedImage [] getBufferedImages();
```

```
}


public interface GameActions{
	/*
	 *@return created game object
	*/
	public GameObject createGameObject();

	/*
	 * update moving objects locations
	*/
	public void updateObjectsLocations();
	/*
	 * it is used to slice fruits located in your swiping region
	 This method can take your swiping region as parameters (they
	 depend on how you calculate it).
	*/
	public void sliceObjects();
	/*
	 *saves the current state of the game
	*/
	public void saveGame();
	/*
```

```
     *loads the last saved state of the game
    */
    public void loadGame();
    /*
     *resets the game to its initial state
    */
    public void ResetGame();


}
```

Good luck 😊