HW03

CSCI: Object-Oriented Programming

Due: 02 Nov, 11:59 pm

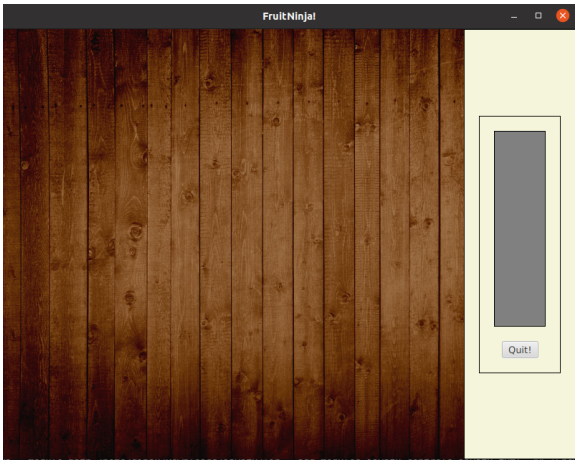
1 Problem Statement

In this assignment, you will write a program that launches a game of FruitNinja with a blade, bombs and four types of fruit.

- You should have a blade and it should have a name to go along with your project.
- The four fruits, namely, Apples (2 points), lemons (2 points), pears (3 points) and peaches (5 points), as well as bombs should randomly launch continuosly once the game starts. These fruits should be washed and ripened before they are launched.
- You should be able to cut all of these fruits with your blade.
  - When a fruit is chopped, the score should increase according to the fruit that you chopped. (Each type of fruit should have a different score). There should also be a splash after it has been chopped.
  - When a bomb is chopped, it should explode and the game should end. Chopping a bomb should not increase the score.
- You should keep track of the score.
  - To do this, you need to initialize the **cs331ScoreController** and have each fruit or bomb keep track of how many points it should be worth when chopped, then passing that value into the appropriate method. You will then use the score controller to update the score throughout the game.
- Look into the FruitNinja Javadocs.

2 Code Incrementally

Step 1: Start by making the **Game** and **cs331ScoreController** show up in the frame.



Step 2: Set up the fruit and generate them in **launchItem()**. Make sure the fruits are washed and ripened when they are launched.

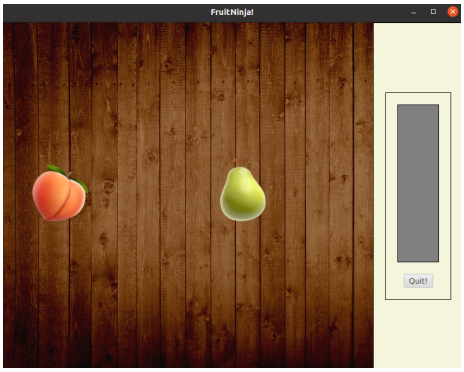


Figure 1: launching objects

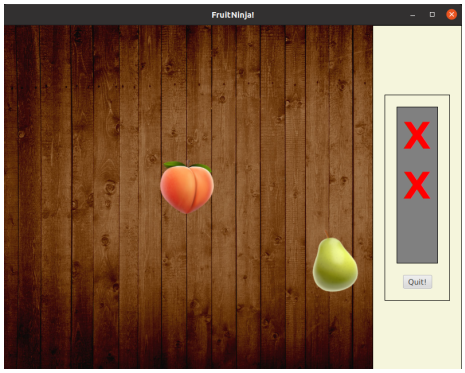
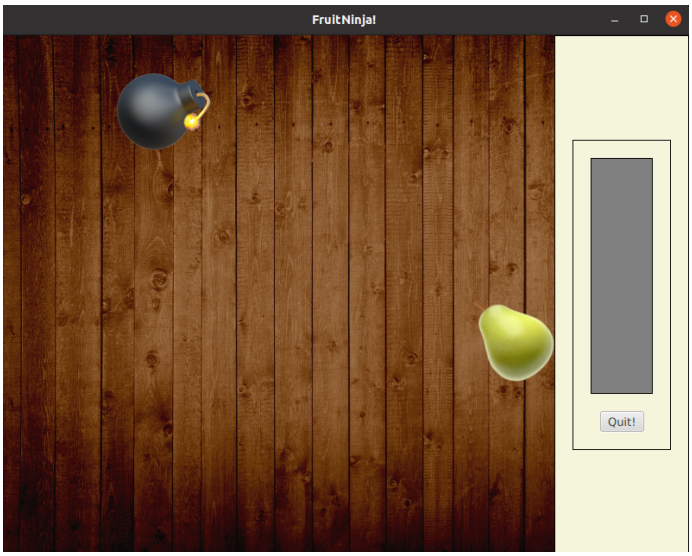


Figure 2: dropped two fruits

- **launchItem** will only generate your fruits *offscreen*. In order to see them move on the screen, make sure to update their position in the **updateChoppable** method by calling the relevant methods. You might have to change **updateChoppable**'s parameter to do this!. You might have to change the launchItem's return type as well.
- launchItem would return Fruit/Bomb object. So you need to create appropriate concrete class for the fruits and/or Bomb, and generate them within this method.
- Make sure you start the game (see **cs331FruitNinjaGame** javadocs) or else **updateChoppable** won't be called.

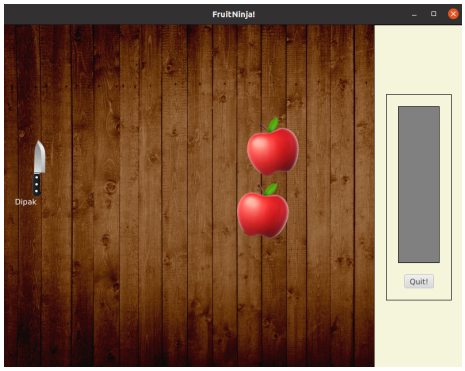
Step 3: Get bombs to appear as well.

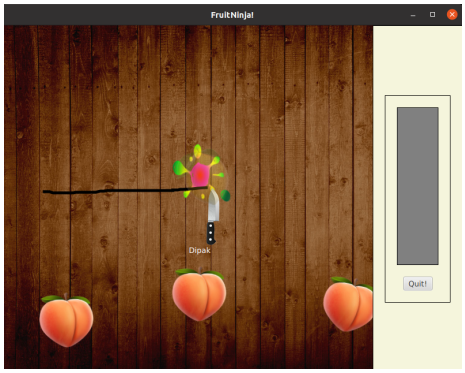
- What could the parameter type of **updateChoppable** become to handle both fruits and bombs.
  Hint: use interface.



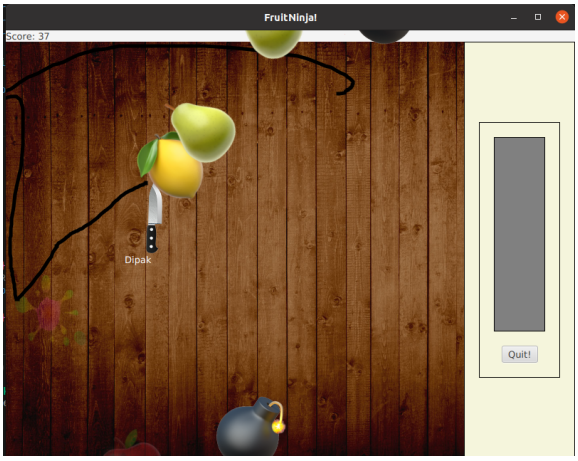
Step 4: Get the blade to appear on the screen.
Implement the fruit and bomb's chopping behavior in the **updateChoppable** method.

- Think about where you should create and add your blade.
- You will want to check if the blade intersects the fruit or bomb that's passed in (see **cs331Blade**). What should happen when a fruit is choppped vs. when a bomb is chopped?





Step 5: Increment the score each time a Fruit is chopped, and show it at the top of the screen.



3 Implementation

3.1 Helper Code

For this assignment, you are provided with three abstract classes, (**cs331Bomb, cs331Fruit, cs331FruitNinjaGame**), each of which should have subclasses that inherit from them. We've provided you with a **Game** class that inherits from **cs331FruitNinjaGame** as an example. You are responsible for writing classes that inherit from the other two abstract helper classes. Remember, you cannot directly instantiate an abstract class.

3.2 Switch Statements

You will need to use a switch statement to randomly launch fruit and bombs onto the screen during your game of FruitNinja. The objects appearing on the screen need to be random, and you might find the **Math.random()** method useful for generating a random number.

3.3 Constants

The helper code's **Constants** class contain static variables you'll likely find useful. For example, to get the filepath to the Apple image, you can use the static String variable **APPLE_PATH**.

4 Design Questions

Think about the following questions while designing your project:

- What do your fruits have in common with each other? What do your fruits not have in common with each other?
- What common actions happen to fruit and bombs and how should they react? (Hint: maybe you should write a short interface, **Choppable** as Fruits and Bombs are both choppable but different things should happen when they move and are chopped.

- Make sure you are not writing repetitive code. Your objects should be generic enough to make adding a different fruit very easy. You might have to alter parts of the program's design and make modifications to your code, however, the more time you spend on the design phase before you begin coding, the fewer changes you will have to make later.

5 Deliverables

Upload your code to the github repo, and share it with me.

Out of curiosity, what's the highest score did you score in the game? Dont' play too much, you have another lab assignment to do as well.