CS102 – Algorithms and Programming II
Programming Assignment 4
Spring 2024

ATTENTION:

- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:
  CS102_Sec1_Asgn4_YourSurname_YourName.zip
- Replace the variables “Sec1”, “YourSurname” and “YourName” with your actual section, surname, and name.
- You may ask questions on Moodle and during your section’s lab.
- Upload the above zip file to Moodle by the deadline (if not, significant points will be taken off). You will get a chance to update and improve your solution by consulting with the TAs and tutors during your section’s lab.

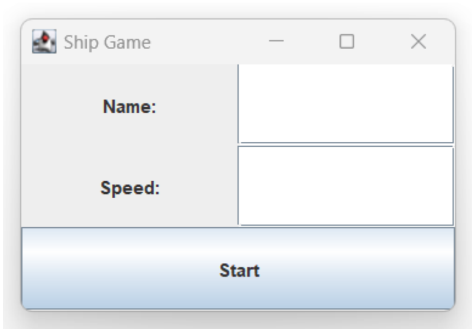
GRADING WARNING:

- Please read the grading criteria provided on Moodle. The work must be done individually. Code sharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The Honor Code specifies what you can and cannot do. Breaking the rules will result in disciplinary action.

A Game of Apples and Bombs

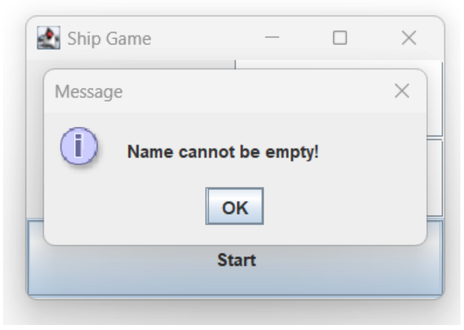
For this assignment, you will implement a simple GUI program where we control a ship to pick up apples in the sea, avoiding bombs. The program should start with a main menu where we enter our name and a speed parameter to begin the game.

You should use the GridLayout (import java.awt.GridLayout) to design the menu. Implement a MenuFrame that extends JFrame. Use a 3 x 1 grid in your frame and add JPanels, each having a 2 x 1 grid layout. Add your JLabel and JTextField elements to the panels to achieve the following look:



Note that you will add your JButton directly to the JFrame so that it uses the whole width of the frame. Thanks to the layout, resizing your window will not disturb the relative arrangement of your GUI elements.

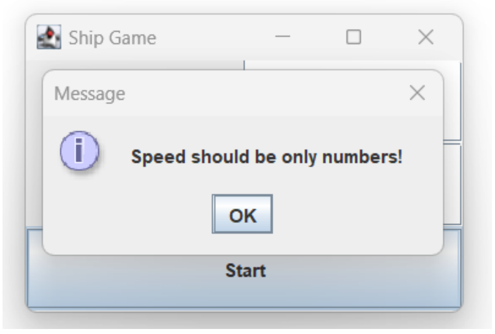
Clicking on the start button should hide this frame and open a different one that displays the game. Ensure the two JTextFields are not empty when the user clicks the start button. Use `JOptionPane.showMessageDialog(this, "Write your warning here");` method to give appropriate warning messages if any of the inputs are empty or the speed input includes non-numeric characters. For this, you may scan the input text and check each character using `Character.isDigit(Char c)` to determine if any character is not a digit.



Ship Game

Message

Name cannot be empty!

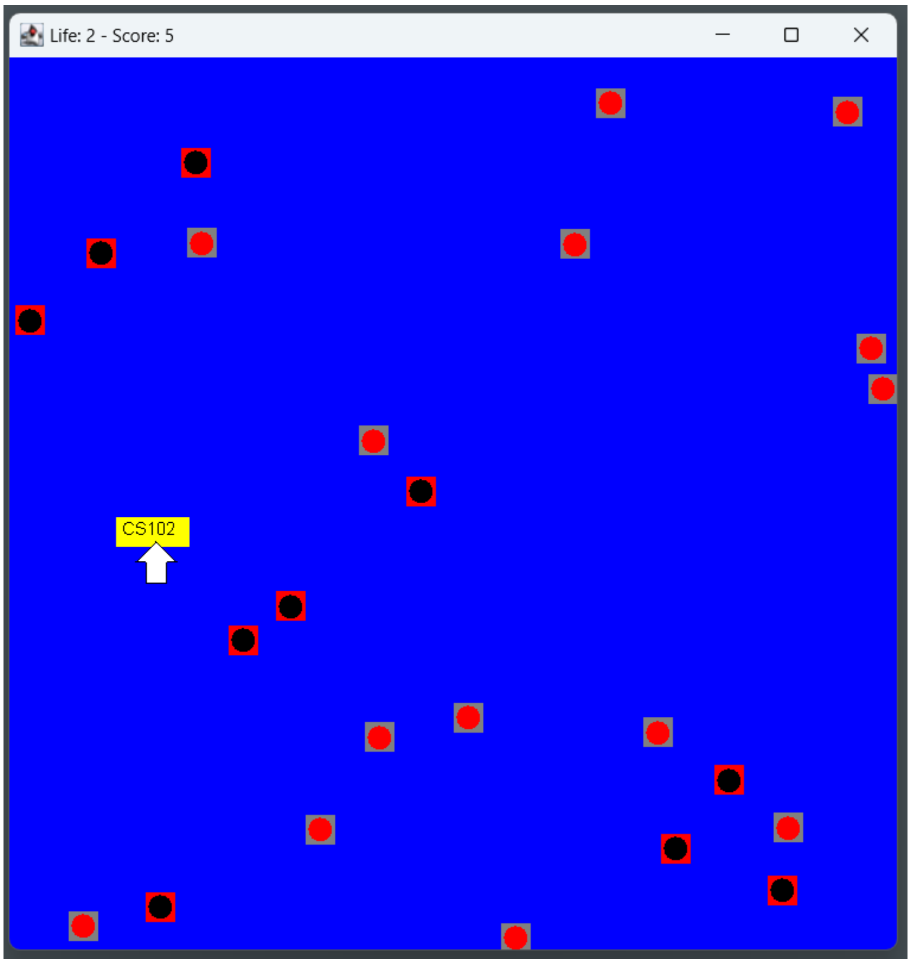


Ship Game

Message

Speed should be only numbers!

Given both inputs are correct, pass them to the constructor of your GameFrame that includes your GamePanel. In this game, we control a ship centered on our mouse location. The ship's location should be updated as we move our mouse within the frame. Some items randomly span on the right edge of the screen and move toward the left: These are either apples or bombs. Each item should cover a square area; you can draw detailed shapes on top of the square area to represent the items better. Draw your ship as a bigger rectangle and write the player's name on top of the ship. A sample view of the game is given below; you may add more details and colors to improve the design.



Life: 2 - Score: 5

In the future, we may include more items that can interact with our ship, so we decide to implement the following interface:

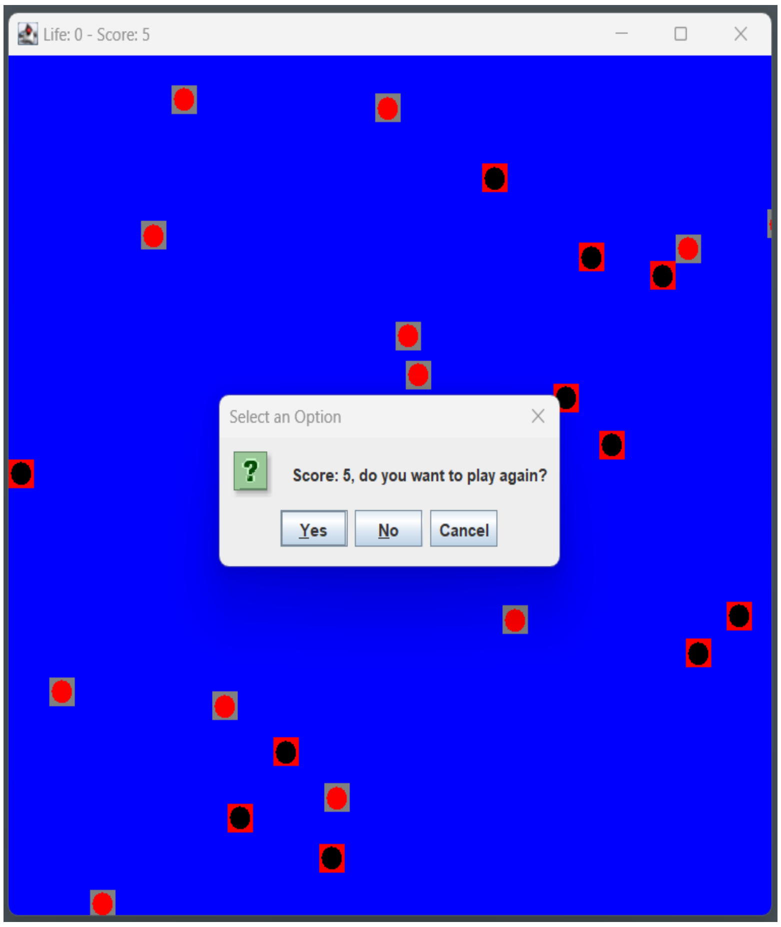
```
import java.awt.Graphics;

public interface InteractableDrawing {
    boolean intersects(Ship s);
    void interact(Ship s);
    boolean moveLeft(int speed);
    void draw(Graphics g);
}
```

Your Bomb and Apple classes should implement the InteractableDrawing interface. You may keep a Rectangle object in your Ship, Bomb, and Apple classes to represent their size and position. Then, you can use the `boolean intersects(Ship r)` method within your bomb and apple classes to simplify the intersection calculations. The method `boolean moveLeft(int speed)` should move the object towards the left edge of the screen with the specified speed amount. You need to pass this speed information from your GameFrame, so GamePanel needs to keep a reference to the GameFrame. A good approach is to pass the reference of your GameFrame to the constructor of your GamePanel using the keyword "`this`". The `moveLeft` method should return true if the object is off limits; in other words, if its right side coordinate is less than zero. Otherwise, it will return false. You will override the `void interact(Ship s)` method to add score to the ship in the Apple class, and to decrease the life of the ship in the Bomb class. Do not forget to add methods to the Ship class to support this functionality. Ship, Bomb, and Apple classes should include `void draw(Graphics g)` method to draw their graphics on screen.

You will use the return value of `moveLeft` method in your GamePanel class to decide which objects to keep and which ones to remove. Your GamePanel class should keep the ship and an ArrayList of InteractableDrawing objects as class members. You will override the `void paintComponent(Graphics g)` method of your GamePanel to draw a blue sea background and draw the ship, bombs, and apples by calling their `draw(Graphics g)` methods. Since bombs & apples need to move toward the left half of the screen, you need to have a Timer (import javax.swing.Timer) in your GamePanel. A Timer should be given an ActionListener whose `actionPerformed` method will be called repeatedly with the specified delay. Your GamePanel class should implement the ActionListener interface to achieve this. Note that the GamePanel class should pass itself to the Timer, not a new instance, so that `actionPerformed` can move the game objects.

As the ship or objects move, their rectangles may intersect. We only care if an InteractableDrawing intersects with the ship. In this case, if it is a bomb, reduce the player's life by 1; if it is an apple, increase the player's score by 1. In both cases, the bomb or apple is removed from the ArrayList. Keep the current life and score of the player on the title bar of your GameFrame. Life starts from 3, and the score starts from 0. Do not forget to update the title bar as the game progresses.

To handle the ship's movement, GamePanel should implement the MouseMotionListener interface; then, you can override the `mouseMoved` method to update the ship's position. Do not forget to repaint the panel and check for intersections when you move the ship. When the player's life becomes 0, display a dialogue using JOptionPane.showConfirmDialog to ask if the player wants to play again. If the player chooses the yes option, restart the game with the same settings; if the player chooses no or cancel, close the game. You are free to add additional methods or classes to support the desired functionality.



Score: 5, do you want to play again?

Preliminary Submission: You will submit an early version of your solution before the final submission. This version should at least include the following:

- The main menu to set the name and speed should be completed.
- You should have the ship moving with the mouse on the GamePanel.

You will have time to complete your solution after you submit your preliminary solution. You can consult the TAs and tutors during the lab. Do not forget to make your final submission at the end.

Even if you finish the assignment in the preliminary submission, you should submit for the final submission on Moodle.

Not completing the preliminary submission on time results in a 50% reduction of this assignment's final grade.