SER 120 – Object-Oriented Programming and Design
Final Project
AVOID THE ENEMIES GAME!

General Instructions:

1- You can work in a group of two

2- You can propose a different project (subject to approval)

3- Read bonus points section

Submissions and due dates:

Due Saturday, 04/20 via email:

- Name of teammate (if you decide to work in a team)
- Project description (if you decide to propose a different project)

Due Monday, 04/29, in class (15 pts see grading rubric):

- Game displays a control panel including all buttons, a drawing panel displaying enemies that bounce off the edges of the screen.

Due Saturday, 05/4, at 11:59 PM on Blackboard:

- • Complete UML class diagram as a PDF, PNG, or JPG file
- • A .zip version of the entire project

Assignment Specifications:

In this assignment you will implement a game in which your avatar will avoid enemies approaching it. Your avatar must be on the right side of the screen and should be able to move up and down. The user must be able to move the avatar both by using the arrow keys on the keyboard and by dragging the avatar with the mouse. The enemies must come from the left of the screen and move straight or diagonally toward the right where the avatar is. The game goes on until an enemy hits your avatar or until the total number of enemies is exhausted. If an enemy disappears from the bottom of the screen without hitting your avatar, you win a point.

The game window should be divided into two parts: the game board and the control panel.

Game board:

The game board is where your avatar and the enemies will be displayed. The drawing board should display a message “Game Over, You win” or “Game over, You lost” at the end of the game.

Control panel:

The control panel will do the following:

1- Allow the user to set the following options *before* the game starts:

- The avatar with which the user wants to play. The user should be able to choose among at least three different avatars
- Total number of enemies
- Difficulty of the game, which will decide the speed at which the enemies come and potentially the number of enemies that come at the same time. You can use three levels: Easy, Normal, and Difficult.

2- Have a start/pause button to start the game and pause it during game play

3- During game play the control panel should display the score. The score should be updated every time an enemy leaves the game panel at the bottom without having hit your avatar

4- The control panel must have a Quit button for quitting the game at any time

Design requirements:

- Your application should include an Enemy class and an Avatar class each representing their corresponding element in the game
- Images should be used for the avatar and the enemies
- The enemies should be stored in an ArrayList
- You must use at least one design pattern, such as Holder or Proxy
- Think carefully about the application design and the relations between the classes you are going to use

Bonus:

There is plenty of room for creativity in this assignment. Here are some ideas:

- Add sounds
- Make it a multi-level game
- Add lives

There will be up to 10 bonus points based on successfully implemented bonus features.

Submission guidelines:

Submit an archived version of your source folder and the UML diagram separately.

Criterion	Points
Program displays a game board and a control panel	3
Game board shows an avatar	3
Game board shows animated enemies	4
Enemies bounce off the edges of the screen	4
Enemies are saved and displayed using an ArrayList	3
If avatar hits one of the enemies or all lives are exhausted, game stops and “Game over” is displayed	3
The user can choose avatar	3
The avatar move correctly using the mouse	3

The score is correctly updated each enemy	2	
The start/pause button works correctly	3	
The quit button works correctly	2	
Images are used correctly	5	
Control panel has option to set game difficulty	3	
Control panel has option to set number of enemies	3	
Program correctly uses at least one design pattern	6	
One anonymous class is implemented	5	
All other specifications have been met	2	
A UML diagram for the program is submitted	1	
The UML diagram contains all classes in the program	2	
The UML diagram shows the proper relationships among classes	4	
Selection of identifiers, comments, and indentation	3	
Program observes naming and visibility conventions	3	
Demo on (demo requirements in bold)	5	
Total:		75
Bonus:		10

Programs that do not compile will not be accepted.