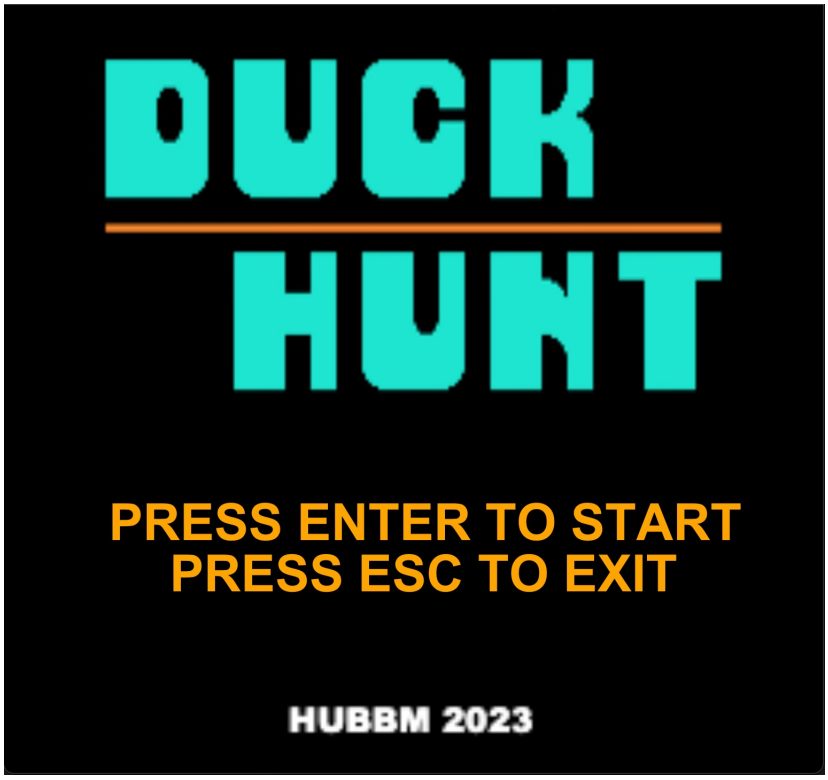
PROJECT ASSIGNMENT 4

Issue Date : 12.05.2023 - Friday
Recitation Date : 12.05.2023 - Friday (15:45) (held on Zoom)
Due Date : 04.06.2023 - Sunday (23:59:59)
Advisor : R.A. Görkem AKYILDIZ
Programming Language : Java 8 (Oracle)
Software Platform : JavaFX



1 Introduction

In this assignment, you are expected to gain practice on developing a Graphical User Interface (GUI) application using Java programming language. As opposed to the command-line programs where the interaction between the user and the computer often relies on a string of text, a GUI program offers a much richer type of interface where the user uses a mouse and keyboard to interact with GUI components such as windows, menus, buttons, checkboxes, text input boxes, scroll bars, and so on. Because most people today interact with their computers exclusively through GUI, developing a GUI based application has become a must for the new developers.
There are many frameworks to develop a GUI application (such as Swing, SWT, AWT, and the like). In this assignment, you are to employ JavaFX framework to complete this assignment. JavaFX is a software platform for creating and delivering desktop applications and Rich Internet Applications (RIAs) that can run across a wide variety of devices. JavaFX supports desktop computers and web browsers on Microsoft Windows, Linux, and macOS.

In this project, a simplified version of game namely Duck Hunt is expected from you to develop through JavaFX framework. All details you need while designing the system are explained in the following section; you will also be given recordings of first two recitations.
Duck Hunt originally is 1984 light gun shooter video game developed and published by Nintendo for the Nintendo Entertainment System (NES) video game console and the Nintendo VS. System arcade hardware. In Duck Hunt, normally player is given some amount of ammo to hit ducks that appears at CRT television with NES Zapper. The ducks can appear in some amount and if the player shoots all of ducks with given amount of ammo, the player will advance to the next round; otherwise, the player will receive a game over. In this project, you are expected to design a version of Duck Hunt game with Java programming language by using JavaFX (Just pure Java code, any other concepts (CSS, FXML, Scene Builder, extra libraries etc.) are forbidden). Moreover, any other assets are forbidden, your program must be successfully compiled and ran with your pure code (*.java files) and given asset files. Note that, contents of asset files can be changed (such as change at sound effects, backgrounds etc.), so please do not code them in static manner, and also please do not submit any asset files as they are going to be given to your code when it is tested.

2 Title Screen

This is the first screen of the game, "assets/effects/Title.mp3" will be played in loop during that screen. User can either go to the background selection screen by pressing "ENTER" key or exit by pressing the "ESC" key. There is a centered flasing text in two lines as follows "PRESS ENTER TO PLAY" and "PRESS ESC TO EXIT" to inform user about what can he/she do to play the game. Title of the game must be "HUBBM Duck Hunt", and also favicon of your game must be "assets/favicon/1.png". Title and favicon must be preserved at each screen and during the game play.

3 Background Selection Screen

This is the options screen of the game, user can either change the background by selecting with left and right arrow keys, and crosshair by selecting with up and down arrow keys. There is a centered text in three lines as follows "USE ARROW KEYS TO NAVIGATE", "PRESS ENTER TO START" and "PRESS ESC TO EXIT" to inform user about what can he/she do to play the game. Moreover, user also can go back to the title screen by pressing the "ESC" key and start the game with selected configurations by pressing the "ENTER" key. Sound effect of the title screen must be still playing without any interrupt during the background selection screen and also it must be still playing if user goes back to title screen. If user presses to "ENTER" and starts to game, sound effect of the title screen gets stopped and "assets/effects/Intro.mp3" gets played for once, after that sound effect ends, game must start from first level. Note that, game must not start until that sound effect finishes. Also note that, options at background selection screen must be reset if some goes to background selection screen again from title screen.

4 Game

Mouse cursor must be changed with selected crosshair icon during the game, note that mouse must be return to system's default cursor icon when user moves his/her mouse to outside of the game window, and also it must return the system's default cursor icon if user returns to the title screen. Game must consist of at least six levels and six directions (going left, right, across the screen -from each corner to its opposite corner-) for the three ducks must be covered. Levels must be sorted according to their difficulty. Ducks must be seen not just as moving but also as flying. Ammo amount for each level must be three times of the duck amount at that level, ammo cannot be transferred to next levels which means it does not matter how many ammo does the user have from previous level, he/she will have just ammo of current level. Ducks must be in between background and foreground, which means if a duck goes to the a foreground object, it will be behind of that instead of being in front of it. Ducks must be reflected in the opposite of the direction that they are coming from whenever they hit to the edges (including corners) of the window (Note that, they are not supposed to be reflected from foreground objects, they just must be reflected whenever they reach the edges of the game window). There must be a centered text at the top of the window that states which level it is and how many total levels there are, sample format for first level of the game which contains six levels as follows, "Level 1/6". Moreover, there must be a text that is at the right corner of the page which states how much ammo is left, sample format for two ammo left is as follows, "Ammo Left: 2". While user shoots (independent from hitting to duck or not) "assets/effects/Gunshot.mp3" must be played once, moreover, if ammo hits more than one duck at same time, it is considered as all the ducks that got hit have been killed, and also ammo are assumed to hit to the ducks that are behind the foreground objects (including floor). Ducks have to fall with animation which starts from "assets/duck_{COLOR_OF_DUCK}/7.png" and followed with "assets/duck_{COLOR_OF_DUCK}/8.png" (images must be flipped if duck is going from right to left when it got hit. Then it has to fall with last mentioned asset. Note that, also "assets/effects/DuckFalls.mp3" must be played once while duck starts **falling**. If user runs out of ammo and at least one of the ducks is still alive, it means that it is game over as loss, so "assets/effects/GameOver.mp3" gets played for once and user gets informed about game is over and he/she can either start again from the first level by pressing the "ENTER" key or go to the title screen by pressing "ESC" key. This information will be done by centered text in three lines (second and third are as flashing) as follows "GAME OVER!", "Press ENTER to play again" and "Press ESC to exit". If user finishes the level successfully (finishing successfully means that killing all the ducks with given amount of ammo), then he/she completes the current level and there are two different situations for completing the level, finishing the last level or finishing any level (which is not last). Finishing the last level will be mentioned later. If user just finishes a level (but not last), as game is not completed yet, just "assets/effects/LevelCompleted.mp3" gets played for once and user gets informed about he/she successfully completed the game. This information will be done by centered text in two lines (second one is as flashing) as follows "YOU WIN!" and "Press ENTER to play next level". So, as informative text says that, user can play the next level by pressing the "ENTER" key, when user presses to "ENTER" key, next level gets loaded. If game is completed (which means completing the last level) "assets/effects/GameCompleted.mp3" gets played for once and user gets informed about game is completed and he/she can either start again from the first level by pressing the "ENTER" key or go to the title screen by pressing "ESC" key. This information will be done by centered text in three lines (second and third

are as flashing) as follows "You have completed the game!", "Press ENTER to play again" and "Press ESC to exit". Sound effects of that level must be immediately stopped if user skips to next level, returns the title screen (title screen's sound effect must be played for sure) etc. Moreover, user must not be able to shoot when game or level is completed or game is over even if he/she has ammo left. Note that you must not play intro sound effect if user just starts the game from end of the game or completing the game just via pressing the "ENTER" key (It means that intro sound effect will not be played if user does not start the game from background selection screen.).

5 Scaling

There must be a scale parameter namely SCALE (as in double format) at main class (which is namely DuckHunt) and it must scale the game. It works as follows, say that the object is x by y in manner of pixel, and scale factor is s, then that object must be s times x by s times y in manner of pixel. Say that object's width is 30, height is 40 and scale factor is 3, then object's width must be 90 and height must be 120 as result. Note that default value for scale factor must be 3 when you submit your work as it is suitable for my working environment, but you can use any other scale factor when you are recording your demo video, it is up to you and your screen resolution. Note that, you can still get partial point if you cannot implement scale as changeable with any global variable but code scale as three by static.

6 Adjusting Volume

There must be a volume parameter namely VOLUME (as in double format) at main class (which is namely DuckHunt) and it must adjust volume of sound effects. It must be in range of [0, 1]. One corresponds to maximum volume where zero corresponds to no sound. For example, if volume value is 0.5, it means that volume is at 50%. Note that default value for volume value must be 0.025 when you submit your work as it is suitable for my working environment, but you can use any other volume value when you are recording your demo video, it is up to you and your sound system. Note that, you can still get partial point if you cannot implement volume value as changeable with any global variable but code volume value as 0.025 by static.

7 Bonus: Horizontal Scrolling

For getting a 15 points of bonus, you must implement a horizontal scrolling (only horizontal, not vertical) mechanism for your game (just for the game, not for the background selection and title screens), the game must start from the center of the whole scene, and user can navigate to left and right by putting mouse icon near to the left and right edges of the screen. If this functionality implemented, the vertical edges must be changed with edge of the map of the game instead of edges of the window while handling ducks reflection at the edges. Note that, please do not break the original concept while trying to add this functionality, the game must work with given rules and this rule is just an extra over them.

8 Restrictions

- Your code must be able to compiled and executed on Java 8 (Oracle).
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- Your code must be clean, do not forget that main method is just driver method that means it is just for making your code fragments to run, not for using them as main container, create classes and methods in necessary situations but use them as required. Moreover, use the four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, Polymorphism) if there is such a need, remember that your code must satisfy Object-Oriented Programming Principles, also you can benefit from exceptions and even if create your own exception class if you need any.
- You are encouraged to use lambda expressions which are introduced with **Java 8**, list structure of Java (and JavaFX) such as LinkedList, ArrayList etc., and your own exception classes.
- You must append a ReadMe file (**"README.md"**) that contains compilation details of your project.
- You must provide a **demo video** (in English with system sounds) that shows requirements. This demo must not exceed 4 minutes (even for one second, and you cannot edit (speed up, slow down, edit, crop etc.) the video for any reason, details of the demo can be reached at "Checklist.docx"). Remember that limit of submit system for each file is 8192 KB, so that you must upload your videos to your YouTube account as **unlisted** and you must keep them as unlisted until the current term (Spring 2023) ends. Place its link to your "Checklist.pdf". You must store all the data about your homework (code, video etc.) until the current term ends.
- You must append a PDF file (**"Checklist.pdf"**) that contains the requirements that you did not satisfy, please state your situation such as "This functionality does not work at all.", "There is a tiny issue such as ..." so that your assignment will be graded by taking them into consideration, please note that if you did not stated a requirement, that is assumed that that functionality is fully working on your system and that will be graded as zero even if a tiny issue occurs that stated as has to be checked (not for the issues that they are not stated at requirements, but remember that there may be some extra conditions that you should check, but they are not going to be graded as zero if you do not satisfy them). Moreover, if you did not provide such a file, it is assumed that your code is working well without issue and that it will be graded as zero even if a tiny issue occurs that stated as must be checked.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- You must use JavaDoc commenting style for this project, and you must give brief information about the challenging parts of your code, do not over comment as it is against clean code approach. Design your comments so that some wants to read your code can

easily understand what is going on. You can check here to access Oracle's own guide about JavaDoc Sytle.

- Do not miss the submission deadline.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as "clean code".
- Use UNDERSTANDABLE names to your variables, classes, and functions regardless of the length. The names of classes, attributes and methods should obey Java naming convention. This expectation will be graded as "coding standards".
- You can ask your questions through course's piazza group, and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Submit system for this homework will be opened a few days before deadline, so please be patient.

9 Execution and Test

Your code must be compiled and executed under **Java 8** (the one provided on Piazza) on Microsoft Windows Operating System (preferably Windows 10 or newer) or Apple MacOS (preferably MacOS 12 or newer) as you will not have any chance to compile and run your program on dev server since it does not contain any graphical user interface. If your code does not compile and execute under **Java 8** by **Oracle**, then you will be graded as 0 for code part even if it works on your own machine under any other versions or providers. **It is forbidden to use any external libraries** such as JSON Library, and **it is also forbidden to use any scene builders, FXML stuff etc.** You are encouraged to build your own GUI just with Java Code as stated at JavaFX slides. It is also forbidden to use any extra CSS stuff. Moreover, your code must use JavaFX for GUI (Swing, AWT etc. is also forbidden). Sample run command is as follows:
Compilation: javac DuckHunt.java
Run: java DuckHunt

10 Grading

Task	Point
Title & Icon	3
Scaling	15 (5 for static)
Adjusting Volume	3 (1 for static)
Keyboard Button Interaction	5
Title Screen	3
Background Selection Screen	7
Changing Mouse Icon with Crosshair	4
Level and Ammo Text	2
Putting Ducks in Between Background and Foreground	5
Flying & Falling Animation	12
Reflection of Ducks at the Edges	10
Sound Effects During Game	9
Detection & Handling of Level Over	9
Starting Game from First Level (At End of the Game)	3
Comments in JavaDoc Style	10*
Horizontal Scrolling (Bonus)	15*
Total	100+15

*** The score of bonus and comments will be multiplied by your overall score (excluding bonus and comments) divided by the maximum score that can be taken from these parts. Say that some got 81 from all parts excluding bonus and comments, 10 from bonus part, and 8 from comments part; then his/her score will be calculated as follows: Multiplied score for bonus part 10 * 81 / 90 which is 9, multiplied score for comments part 8 * 81 / 90 which is 7.2, and overall score of his/her is summation of 80, 9, and 7.2, which is 96.2.**

10.1 Score Multipliers

There will be five major overall multipliers which are going to be scaled over 1 and they will be multiplied by your grade, they are Correct Submit Hierarchy Multiplier, OOP Multiplier, Demo Multiplier, Game Rules Multiplier, Graphical Design Multiplier.

- Correct Submit Hierarchy Multiplier: If you do not obey the submit hierarchy (you have to score exactly 1 point at submit system, otherwise it will not be accepted as correct hierarchy even if your hierarchy fits to format), you will be penalized with 20% of point deduction.
- OOP Multiplier: If you do not obey to OOP at all you will be graded as 0, and also there will be some values for OOP multipler between 0 and 1 with respect to your design. So, OOP Design is as important as resulting with a correctly working game as you may get 0 from this project if you will not obey the OOP rules at all, even if your code works fully correct.
- Demo Multiplier: If you do not provide any demo video or if it is not in English, demo

multiplier will be 0. Also partial deductions at multiplier are possible if there are missing parts at demo.

- Game Rules Multiplier: If you do not obey the given rule set of the game (such as default ammo count, minimum level count etc.), game rules multiplier will be deducted from 1 towards 0.
- Graphical Design Multiplier: If your GUI design is not as desired, you will be penalized by deduction of graphical design multiplier from 1 towards 0.

11 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only not compressed .zip files because the system just supports .zip files).

```
- b<studentid>.zip
    - Checklist.pdf
    - README.md
    - <src>
       - DuckHunt.java
       - *.java
```

12 Late Policy

You have two days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You must submit your solution in at the most two days later than submission date, otherwise it will not be evaluated. Please do not e-mail to me even if you miss the deadline for a few seconds due to your own fault as it would be unfair for your friends, e-mail submissions will not be considered if you do not have a valid issue.