Yanjun and Jonas
CMPT 276 Assignment 4
April 11, 2024

Yanjun:

-Large Class: “GamePanel” class contained an excessive number of variables and methods, making it a "God Object" — a class that does too much.

-Refactor Step:

-Extract Interface: Created an interface named ScreenSetting that encapsulates all final variables related to game screen settings such as tileSize, screenWidth, and screenHeight.

-Extract Abstract Class: Developed an abstract class to house common properties and methods for setting up a game panel. This includes initialization, getters for hero, enemy, and items, and foundational game methods like startGame and endGame.

-Inheritance Hierarchy Refactoring: Refactored MainGamePanel to inherit from the new abstract game panel class, promoting code reuse and organizational clarity.

-Improvement:

-The GamePanel class now has a more focused role, with screen settings abstracted to an interface and shared functionality moved to an abstract class. This makes the system more modular and easier to manage.

-Commit link: ff41e48158b3c19bcb98b886e4b0e1a316069476

Duplicate Code: The “setImage” method was duplicated across multiple classes that handle characters and items. This duplication leads to increased maintenance effort and potential for inconsistencies.

-Refactor Step:

-Extract Method: Created a new method in a class named UtilityTool to handle the loading and setting of images from resources.

-Utilize Inheritance/Composition: Refactored character and item classes to use an instance of UtilityTool for image setting tasks, thereby removing the duplicate setImage methods from these classes.

-Improvement:

-The refactoring has eliminated the duplicate setImage methods across various classes, centralizing the image loading logic within a single UtilityTool class. By using the UtilityTool class, we promote code reuse, making the overall codebase cleaner, more maintainable, and easier to understand.

-Commit link: 6e6d87bdaa8f422afd3910065662395441c2ed87
5eba9f6787552aeb6b35d2f4258a1fc6bb60e611

Long Method: The “update” method was too lengthy, making it hard to understand and maintain. It included several responsibilities, like handling collisions, which could be abstracted out.

-Refactor Step:

-Extract Method: Split the collision-related responsibilities into separate methods - “checkTileCollisionAndMoveHero”, “checkEnemyCollision”, and “checkItemCollision”.

-Create Wrapper Method: Developed a new method “collisionCheck” to encapsulate the three new methods related to collision checking.

-Replace Code with Method Call: Substituted the collision-related code in the update method with a single call to the new collisionCheck method.

-Improvement:

-The update method's complexity is significantly reduced by abstracting out the collision detection and handling into separate methods. This makes the update method shorter and more readable, adhering to the Single Responsibility Principle. It also makes the code easier to test, as collision logic is now isolated and can be tested independently.

-Commit link: a2b481a9376def79960467d69c411c7d9b758f5d
cd127434e21c0898ffdcc0ff4868245c894f4396

-Duplicate code/switch statement: In the “draw” method within the abstract Character class. The assignment of currentImage occurs multiple times and the switch statement overuses for a simple condition.

-Refactor Step:

-Extract method: Isolated the logic for setting currentImage into a new method updateCurrentImage.

-Replace Conditional with Polymorphism: Use a ternary operation instead of switch for simplicity.

-Improvement:

The draw method now solely focuses on drawing, and the logic for determining the image has been abstracted to another method. Also, eliminating the switch statement simplifies the control flow, making the code less prone to errors during future modifications.

-Commit link: 654dac655bc953beb39ba2249b4170652c2d8d99

Jonas:

- Duplicate code in UI class: line 80 and 90 redefines a in the iteration of draw, this was presumably done in original coding because ‘b’ was updated and modified slightly to ensure the end of game message displaying the congratulation message, time and score would be spaced out evenly. But this is only applicable to the y axis, not the x axis (it is all centered based on our design)
- Refactor step: Comment out (or just remove) the code, since this will not affect anything.
- Improvement: Small performance optimization gained since each time the draw method is called these two lines are no longer there to be iterated over.
- ==========
- Unused variables (I guess?) in the UI class: on line 110 wordsTime is > 45, with 45 being an arbitrary value chosen during development, which represents how long a message stays on the screen for (with 60 being 1 second).
- Refactor step: I have changed this number that seems hard coded to a private static final constant above called MAX_WORDSTIME to improve readability and in case it needs to be called again, it can be applied consistently. Also, if in the future we decide to change this value, changing the constant at the top will suffice, instead of manually going through the whole code and finding every instance that’s applicable.
- Improvement: Improving readability and convenience for future editing and coding/development.
- ==========
- Bad/confusing variable names: I named the graphics in the draw function gr2D during development to highlight the fact that it was 2D graphics and not just graphics, but this just ended up being unintuitive and awkward to type over and over again.
- Refactor step: changed the name to graphics to remove ambiguity.
- Improvement: easier on the eyes and fellow project teammates (including myself) so I don’t have to take a few minutes figuring out what gr2D is.
- ==========
- Lack of documentation / poorly structured code: In the public UI method on line 35, a new Font object was created for a40, a60, and a80BIG. This can be improved!
- Refactor step: Modify the code logic so it calls a separate function, namely createFont to abstract and structure the creation of Fonts to make more sense.
- Improvement: performance boost and no need to ‘new Font’ each time a new font is made, especially if the game will be expanded to include way more different fonts.

All my commits were done in one push titled update UI.java: 6c624d1c7c1feaac690a2607e1b66f3e011901e6