Java Project 2B – Reflection David Pham 756598

/////////////////////////////////////////////////// Preface //////////////////////////////////////////////

The project was a game. Doing it was no game.

Having been involved in a university engineering competition, I hoped my extension would give me some space.

It turns out that this project would be the most thrilling and stressful and tumultuous experience of my computer science assignments thus far, but probably more because I only had 4 days to do it.

Nonetheless it has been the most effective and concentrated learning experience of Java that I could possibly have received and although it was a struggle, I finally have something new I can use to procrastinate over!

Now to the code!

// //////////////////////////////////// Major changes since Project 2A /////////////////////////////////

I have made several changes to my class structures since my submission in Project 2A they include:

* Removal of unnecessary interfaces being: movable, pushable, destroyable and toggleable.
* Removal of unneccessary class Tile
* Removal of array of Blocked positions from code -> each sprite holds that information itself
* Refinement of the block, player and enemy interactions.
* Functionality added to classes to Position and sprite
* A time attributes to Skeleton, Explosion and Ice

//////////////////////////////////////////////// Special Changes ////////////////////////////////////////

// - Removal of redundant Interfaces and Classes

Although Project 2A was meant to be the blueprint for how my code was to be set out, it seemed to be more of a means of understanding the assosciations and relationships between the following objects. Coding becomes more meticulous the more classes I added and Project 2B was finding a balance between giving each class their identities but still making it flexible enough to edit.

The Tile abstract class which helped me group similar classes was more of a mental assosciation, but it added no extra functionality so was removed.

Interfaces like movable, pushable, destroyable and toggleable helped me group similar behaviour but as my classes began to be more distinct, it became unnecessarily inflexible to incorporate.

Inspired from the example solution to Project 2A, I felt inheritance extensions through classes such as Enemy and Block was all that necessary to characterise the following objects.

// - Block Interaction

To simplify how my current game handles block dynamics:

* ‘Blocks can be asked if they can move before something moves into a block’s position.’
* Later down the world class update()
* If a block saw a player or Rogue in its space
* It would (politely) move in the same direction that the Player or Rogue had moved to get there. (Yes for a spilt moment they would occupy the same space)

Upon initial consideration for how to handle this situation, I *pushed* myself into believing that the only means was by getting the block to move out of the way first.

But when I changed my data management removing my arrray of all blocked locations and let each sprite hold their own “roadblock” information, this allowed me to implement the new above mentioned code which helped me separate the “check” and “relocate” phases assosciated with movement.

(Ohey I’ve hit the 500 word requirement)

// - Class ammendments

Still being very knew to multi-class programming, I was still learning the *ins and outs* of how data was passed between objects while battling the “multiheaded beast” that was privacy modifiers.

It was through coding that reinforced when getters and setters were necessary and where to use “this”, “super” and become more comfortable moving away from static variables.

My Project 1 stored the single level as a static object making it easier to access and refer to, but over time I began making objects (namely of position and world Class) pass themselves through their methods for functionality.

It was during this process that I developed a greater understanding of how objects would not only pass but modify each others attributes.

///////////////////////////////////////////////// Ending ////////////////////////////////////////////////

If I were to change what I did during this project I would better plan how to accomplish and stabilise small segments as opposed to creating testing all the classes at once.

During my time constraints this felt impossible to do as sooner than later I will have had to finish everything.

In doing so I would have had more confidence that what I had in my head also worked on the computer.

Thank you for taking the time to read my reflection and hopefully it helped you gain a better understanding of how my code works!