DYEHARD REFLECTION README

Author: CHUAN WANG

Contact emails:

[Chuan.wang101@gmail.com](mailto:Chuan.wang101@gmail.com)

[kyodivine@qq.com](mailto:kyodivine@qq.com)

How Reflection Works:

Student will be able to make a StudentObj.java Class.

StudentObjManager.java will validate it and such, validate() method has more details on requirement, this can be changed based on different requirement. Validate() currently returns 1 boolean, this can be later changed to return a hash map of strings(method) to Boolean, for further control. The class calling the Java.Reflection is in ClassReflector.java

The validate() method in StudentObjManager can be changed to take in a string to allow student to make whatever name they want, or if its deemed necessary for them to make multiple Object Classes.

Once validated, useStudentObj Boolean variable will be set to true to use the studentObj

The student can then register objects as either a hero or an enemy. As of now, it only allows one instance of hero. Enemies can be multiple cases.

Upon registration, StudentObjManager will have a reference to both the StudentObj and the GameObject in a hashMap, the student/teacher driver will have a reference as well. Currently the teacher driver has reference to the GameObject Hero as well to be able to call fire(), this can be avoided by just calling fire() in the game code instead.

The teacher driver is very simple at this point, no fps issues, I have tried it in the full game, the fps drop is unnoticeable on my machine in window-mode. (full screen has fps issues)

The student/teacher driver class (usercode) will make changes to the StudentObject, and in StudentObjManager will check for changes per frame and update the GameObject based on changes.

StudentObjManager is inside the DyeHardAPI/src/dyehard folder.

Student/Teacher driver is UserCode inside DyeHardSample.

The actual game itself has issue with memory heap being too large. I got around it currently by spawning a dummy process in Main (DyeHardUser), and setting the Java Heap with –xmx using .exec. Current implementation requires the exported jar to be named DyeHard.jar (for full game only, not for Sample driver)

This is bad solution, if memory issue is ever solved by using updated driver, main should only contain:

(new Main()).startProgram();

NOTE\* all code are inside the JAVA folder

Reflection Diagram:

