Maxwell Bryant
3/10/16

Slick Summative Essay

Computer Science is fun yet grueling. This Slick based game really kicked me in the teeth, but it also was exciting to fix problems and know that you did everything; the glory belonged to me. That was at least what I thought before my grade was borderline failing. I guess my creativity blocked out what I actually needed to do. However, when the dust settles, I will have become a better programmer.

Now, to the good stuff, I will be explaining my game to the people of America! First, why was my game divided into classes? Well, random thoughts, my game, and everyone else's classes were divided for organization. This indubitably, is one of the reasons, but not all of them! Another reason for this is efficiency; you don't want to be looking around for something all day, so Java did something nice allowing someone to split the program from its classes and put them elsewhere to be called. Following this question up, objects and variables are often accessed from their respective splitted classes so that the objects cannot be accessed anywhere else. They are exclusive V.I.P. things that only go to the parties they are invited to. In this case, if an object lived in a "Stuff" class and the object was a "skateboard", you could call for a skateboard in the stuff class to be at the park; in this case, your main class.

Well, now we are on question three… it's sad I've been babbling on for this long. Anyway, the uses of public, private, and static modifiers are essential to a great game. Private obviously keeps the value between the class and the object and/or whatever you are modifying. The public modifier can be passed to the main class easily if the class has been split off. Finally, the static modifier, it does everything! You can pass it, change it, live in it; the static modifier can be used indefinitely. If it was an object, it would be a dollar bill. Now, we go onto Arrays. I used Arrays in my game when I was creating my prison bars for the keys. Instead of making two individual bars that would be in about the same place. I made an Array of bars, and saved at least an hour of my time. How I did this was simple: String[] bars = [bar1, bar2, bar3, bar4, bar5, bar6, bar7, bar8, bar9, bar10];

I used ArrayLists for the keys! They have the same stuff in them except for when you had to do extra code to make them unique. Here is an example of what I did: public Keys yk, bk, pk, gk, rk; then… public ArrayList<Keys> keyz = new ArrayList();. My primary objects were the bars and keys as I discussed earlier. I made them effectively with Arrays and ArrayLists, but then I gave the bars their individual coordinates and the keys their unique qualities. Here is an example of how I set the key coordinates in the initialize class: key1 = new Red_Mark(6 * 64, 6 * 64); then… keyz.add(key1);. The roles of these objects are relatively simple. You pick up the correct key, and walk into the correctly marked bar. Then, you get the next key. If you didn't have the proper key, it would not let you through.

Objects, they are a thing in your hand… except when it isn't. In this case, the programmable objects are quite different from the actual things. To define an object in code, it is a class that contains class constructors giving the class a meaning. An example of this is in one of my key classes. They are almost all the same, but they all construct what size they are, and what pictures they grab from the "res" folder. Now, onto a more pressing issue, why does Mr. Davis make us have a "player" object and a "Player" class? Well, the world may never know, but I can tell you why the object player was called with static values! If we made the "Player" class static, it would've shared all of its values with the program! This is a big no because if you had an enemy, they would clash and mess up! This is why Mr. Davis had us make an object "player"; it is able to be used without affecting the class in the wrong way. For example, I would use the "player" object to call the sprite animations into the main class.

Finally, I get asked a personal question, what have YOU learned from Java and this game? Well, I have learned quite a bit from this experience. For starters, I learned what Object Oriented Programming was and how it worked. All in all, I think I will be taking a lot of this information to when I plan to start a company. Java these days is incredibly useful and it is essential to know so you aren't paying some foreign guy to do it for you. Therefore, I can officially cross "make a game" off my bucket list; that is what I have learned.

BONUS: I think that continuing to make and create games will expand my knowledge on general Java and broadens my spectrum for colleges to look at. For example, if colleges were to ask, “What have you done besides engineering at ETA?” I would simply respond, “Well, I was in a Computer Science class and when we built a game, I felt inspired to continue the joy that is produced from creation.” I simple man’s terms: I KNOW HOW IT FEELS TO BE A GOD. Therefore, this is why I will strive to keep Java and Computer Science in my life, even if I don’t have the class.