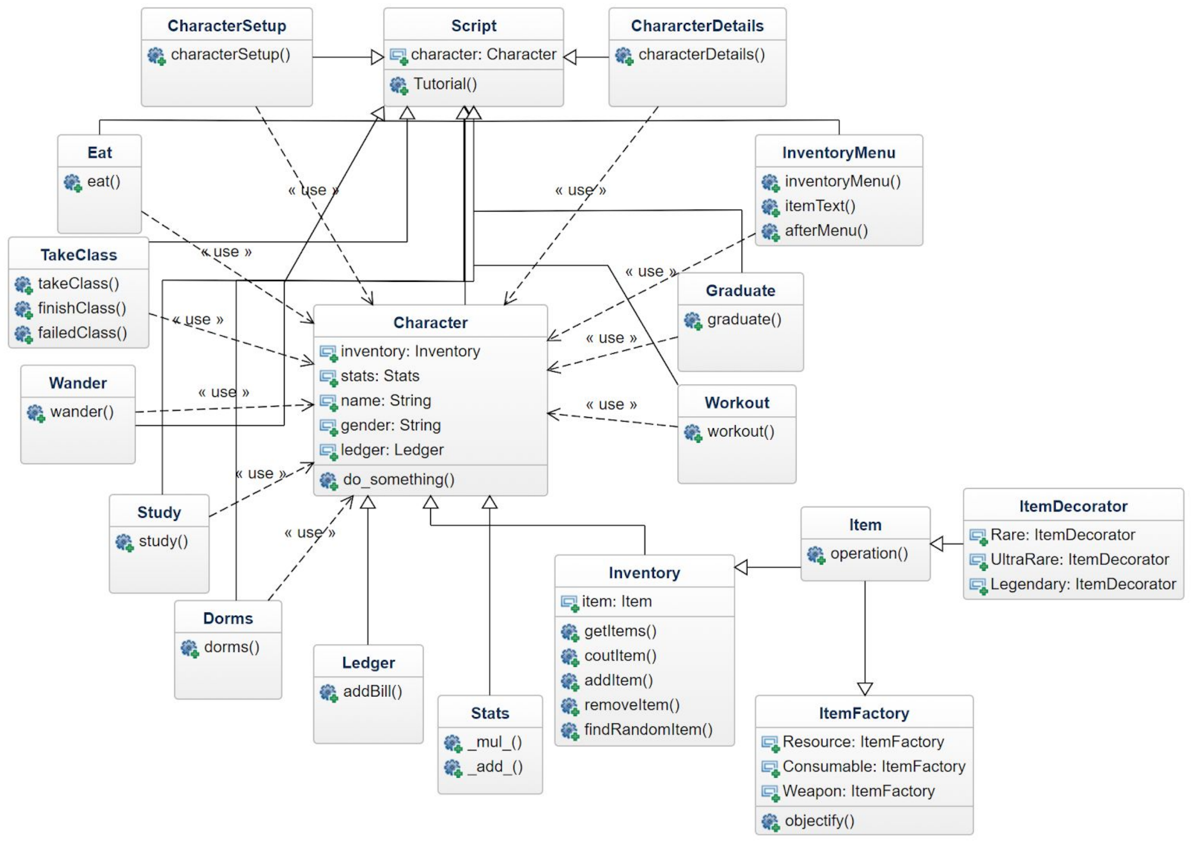
Sid Bostwick
Jackson Lee
Ana Vukojevic

Gates of Galloo

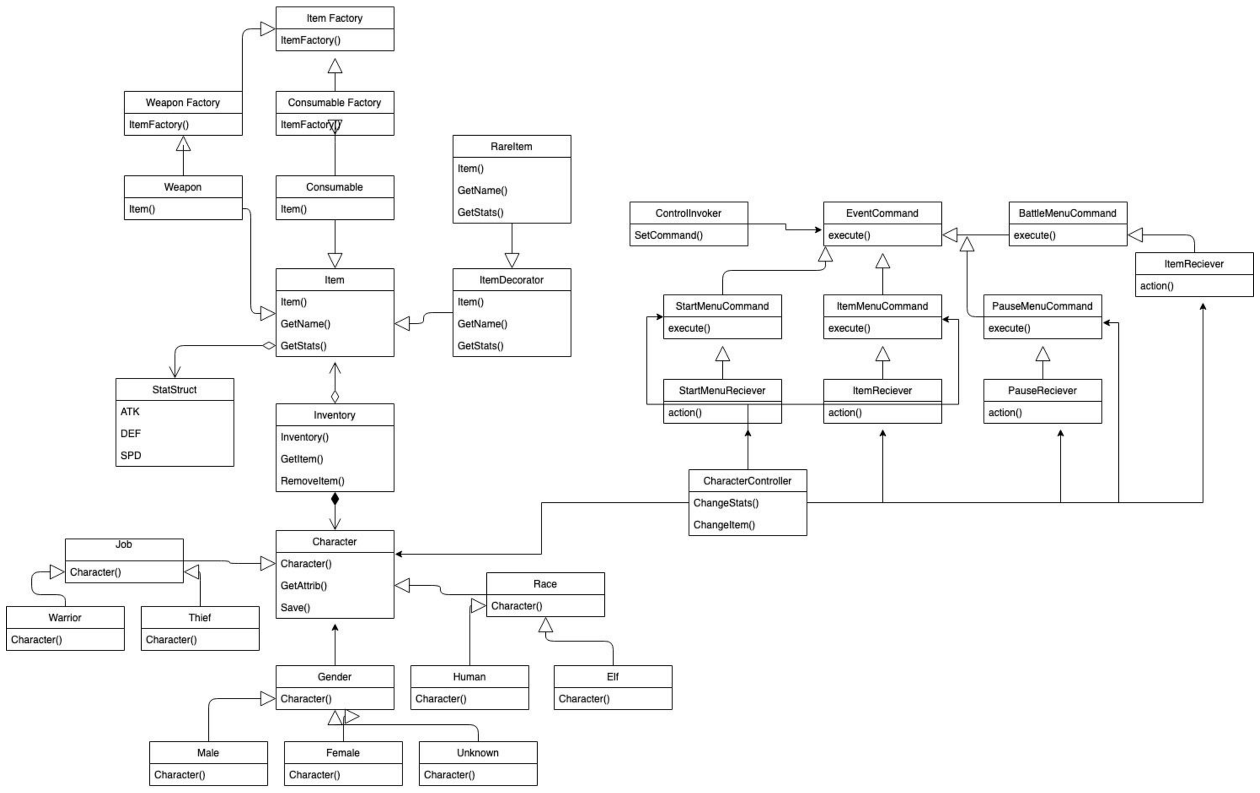
Final State of System

All functional and non-functional requirements outlined in the project design were achieved. The major change made to the project was the shift in the game's contents, it is no longer a fantasy adventure game, but instead an adventure game centered around college life. This resulted in some changes to the class structure in relatively minor ways– instead of choosing a race for your character you choose a major– and other similar alterations. The core functionality remains the same, with the battle system now applied to coursework, where studying and working out can increase your stats and, as a result, your chances of success in the course.

Final Class Diagram



Previous class diagram:



The key changes in the systems shown in the class diagram is the shift to the new storyline ('Race' and 'Job' were implemented but are used in the end to represent student type or for differentiating between student and professor character types) and the changes made to accommodate the system used, RenPy. The EventCommand was only implemented for the battle system, the start menu, item menu and pause menu all were much easier and cleaner to implement using the existing RenPy interface. The character class was also modified slightly to have more attributes to be more fitting to our game, adding attributes such as ledger to reflect the balance owed to the university and an itemized list of each payment. The ledger however is simply an array of Bill objects. At its core, the class structure for the project remained mostly the same.

Third Party Code

The third party code used in our project consisted of:
Obviously the most important third party code that was included was RenPy which provided the framework for the game in the form of a Python virtual machine. The VM libraries are precompiled into cython scripts for each individual system distribution making it somewhat more challenging than simply using pip to install additional libraries. On Linux however one can simply open the VM and do exactly that, Mac and Windows are the more affected operating systems. The engine provides native hooks for game specific functionality including calling Python scripts directly from the script parser, for this reason half of our code is in .rpy files. The rpy file scripting format is more or less the same as pure Python.
https://www.renpy.org/latest.html

Millify python library for human readable numbers
https://github.com/azaitsev/millify
The github repository also contains PyDBlite as a pure Pythonic library for database creation, although this was ultimately unused since it didn't add to the object orientedness of the project, and saving was achieved more easily through cPickle.
https://pydblite.readthedocs.io/en/latest/

OOAD Process

A key design process element we used was CVA. Finding the commonalities and variations in our project allowed us to structure our classes in such a way that we could use base classes to implement the multiple variations of certain classes, thus simplifying our class structure and code quite a bit. We also faced the common design process issue of changing requirements. As we altered the storyline for our game, we found that some of the requirements and class structures also changed, and we were forced to make some changes to existing code. We also got to experience in-depth the importance and use of elements such as UML diagrams and use cases. We found that by spending time on class diagrams and use cases, we greatly simplified the programming process and increased our efficiency.

Original Code

Under the game folder, all of the .rpy files with the exception of tutorial_playing.rpy were fully implemented by our team. Under the game/Python folder, all of the files with the exception of Millify.py were also implemented by our team.