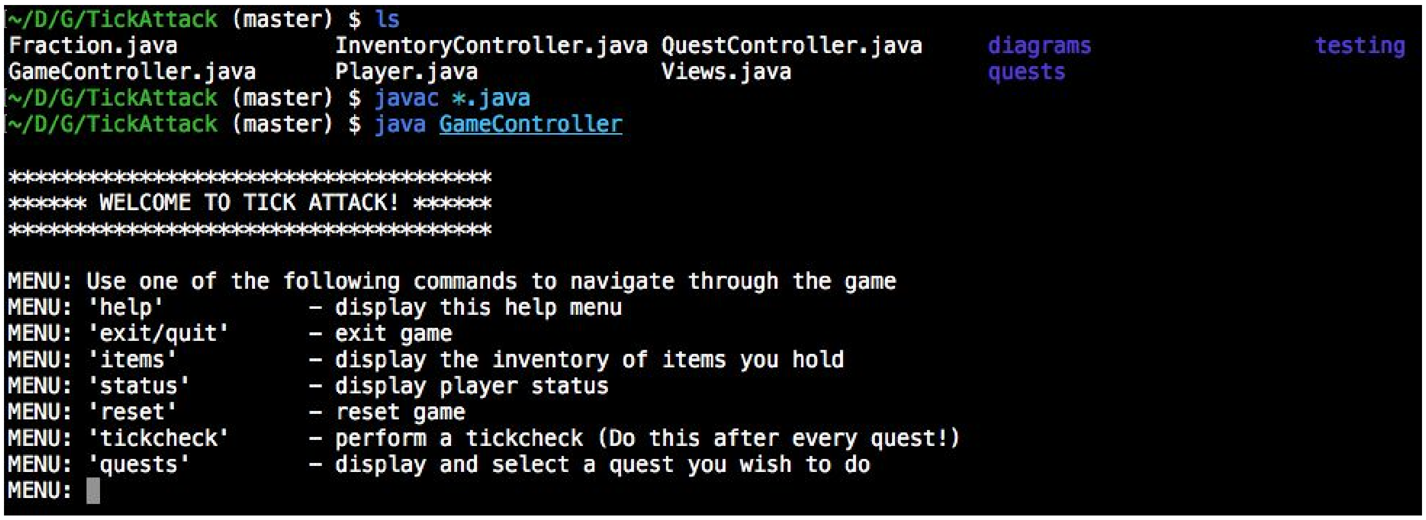
Jonathan Whitaker
William Fiset
COMP 3721

Milestone 3 - Tick Attack Game

How to run game

You can begin by running the game by compiling all the java files and then executing the GameController which contains the main method:

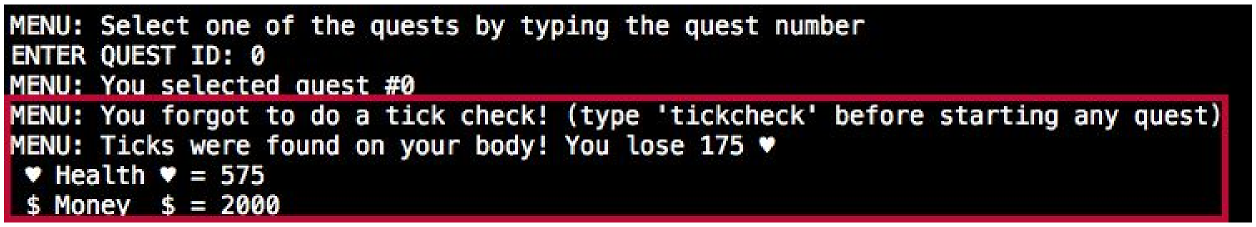


```
~/D/G/TickAttack (master) $ ls
Fraction.java            InventoryController.java QuestController.java     diagrams                 testing
GameController.java      Player.java              Views.java               quests
~/D/G/TickAttack (master) $ javac *.java
~/D/G/TickAttack (master) $ java GameController

*************************************
****** WELCOME TO TICK ATTACK! ******
*************************************

MENU: Use one of the following commands to navigate through the game
MENU: 'help'           - display this help menu
MENU: 'exit/quit'      - exit game
MENU: 'items'          - display the inventory of items you hold
MENU: 'status'         - display player status
MENU: 'reset'          - reset game
MENU: 'tickcheck'      - perform a tickcheck (Do this after every quest!)
MENU: 'quests'         - display and select a quest you wish to do
MENU: 
```

How to play game

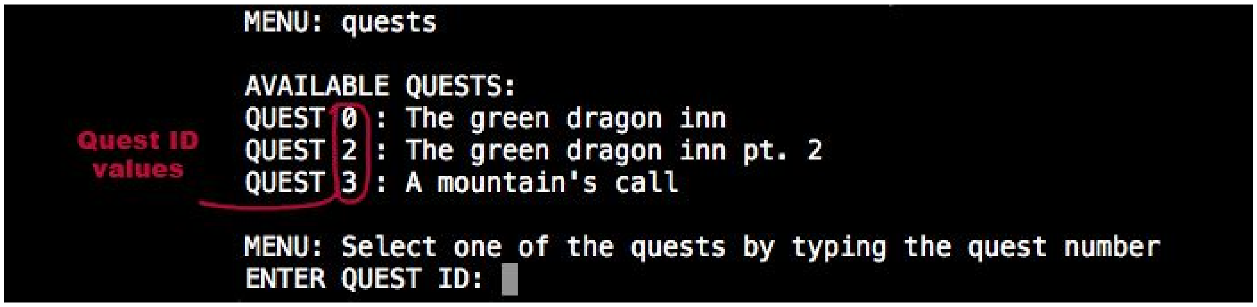
The first thing that happens when the game is launched is the help menu is displayed and gives a synopsis of all the valid commands the user can enter. Most of these commands are self explanatory, so let's cover to less intuitive ones:

'tickcheck' command: The tickcheck command is probably the most important of all the commands you can use. As you go on adventures you acquire ticks, so it is important that after you do ANY quest that you perform a tickcheck otherwise you will lose health for not doing so and ultimately die. This game feature was designed to get players to remember that after they go outside (in real life) that they should check themselves for ticks. If you forget to do a tickcheck in the game them you will get a message like the following:



```
MENU: Select one of the quests by typing the quest number
ENTER QUEST ID: 0
MENU: You selected quest #0
MENU: You forgot to do a tick check! (type 'tickcheck' before starting any quest)
MENU: Ticks were found on your body! You lose 175 ♥
 ♥ Health ♥ = 575
 $ Money  $ = 2000
```

'quests' command: The quests command allows you to start new quests. By typing in 'quests' a menu of available quests will appear and ask you to type in a quest id:



```
MENU: quests

AVAILABLE QUESTS:
QUEST 0 : The green dragon inn
QUEST 2 : The green dragon inn pt. 2
QUEST 3 : A mountain's call

MENU: Select one of the quests by typing the quest number
ENTER QUEST ID: 
```

If you type in a valid quest ID value the given quest will start.

Adventuring through quests in our game design is very fun because different events along a quest happen with different probabilities. That being said, it is possible that when doing the same quest more than once different outcomes will happen. When running the same quest more than once it is entirely possible that you unlock new quests and acquire items you have never encountered before.

MVC Design

Our project was designed to adhere to the Model View Controller (MVC) paradigm even though we were designing a text based game in the terminal without any GUI features. If in the future we do decide to move towards using GUI framework we would have no trouble adapting our software to fit the new requirements.

Models: We have two primary models that we use in our implementation of TickAttack. These are the inventory model and graph model.

1) Inventory Model: The inventory model contains Items with descriptions. The inventory model lets you add, remove and iterate over items in your inventory.

When the inventory controller is invoked it requests access to the items inside the inventory model and can then display the items

2) Graph Model: The graph model forms a basis for how we can navigate through quests in our game. A graph model lets us construct a graph representation of the various events that happen in our game regardless what they are. An essential functionality of the graph model is its ability to tell the controller which event is to happen next. In our game design different events happen at different set probabilities so it is important to be able to randomly select a next event given where you are in the graph.

Controllers: We have one primary controller in our game which is the GameController and its two child controllers (QuestController and the InventoryController) which get invoked by the parent controller as needed.

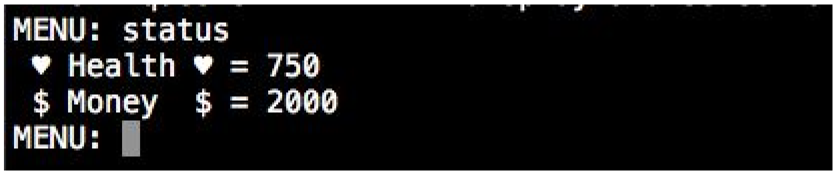
1) GameController: The GameController is the main brains of the program. It invigilates all the action that goes on and responds to user input commands. The game controller is also responsible for invoking the quest controller if the user wants to do a quest or the inventory controller if the user wants to view the items they have collected.

2) QuestController: The quest controller's job is to play a quest. Given a valid quest id number the quest controller will walk you through the events that happen in that quest.

3) InventoryController: The inventory controller is the simplest of all the controllers, it's only job is to display the items in the player's inventory.

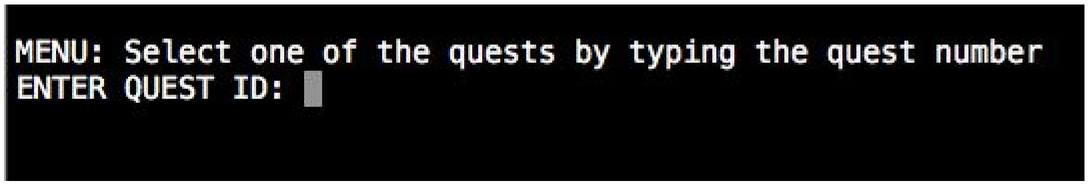
Views: Originally we only had one view in our application. All this view could do was display text to the screen and read input from the terminal. However, we soon realized that even for a simple text based game there were still "views" that we could use.

1) Generic View: This is the original bare bones view. All it can do is display lines to the console and read input from the console. This view serves as the base class for all the other views.



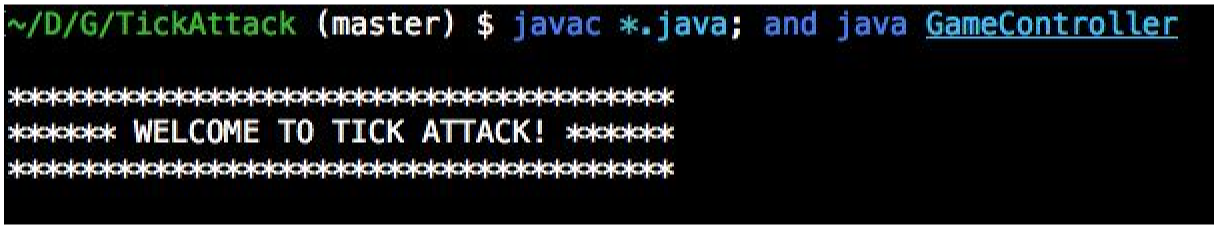
```
MENU: status
 ♥ Health ♥ = 750
 $ Money  $ = 2000
MENU: 
```

2) Header View: The header view is the most used type of view in our game. All the header view does is it displays some header text before displaying the actual desired text. Usually the header will be one of the following "MENU", "QUEST" or "INVENTORY" to indicate which section of the game you are in. Here is an image of a header view asking the user to type in a quest number and another header view below waiting for the user to type some input:



```
MENU: Select one of the quests by typing the quest number
ENTER QUEST ID: 
```

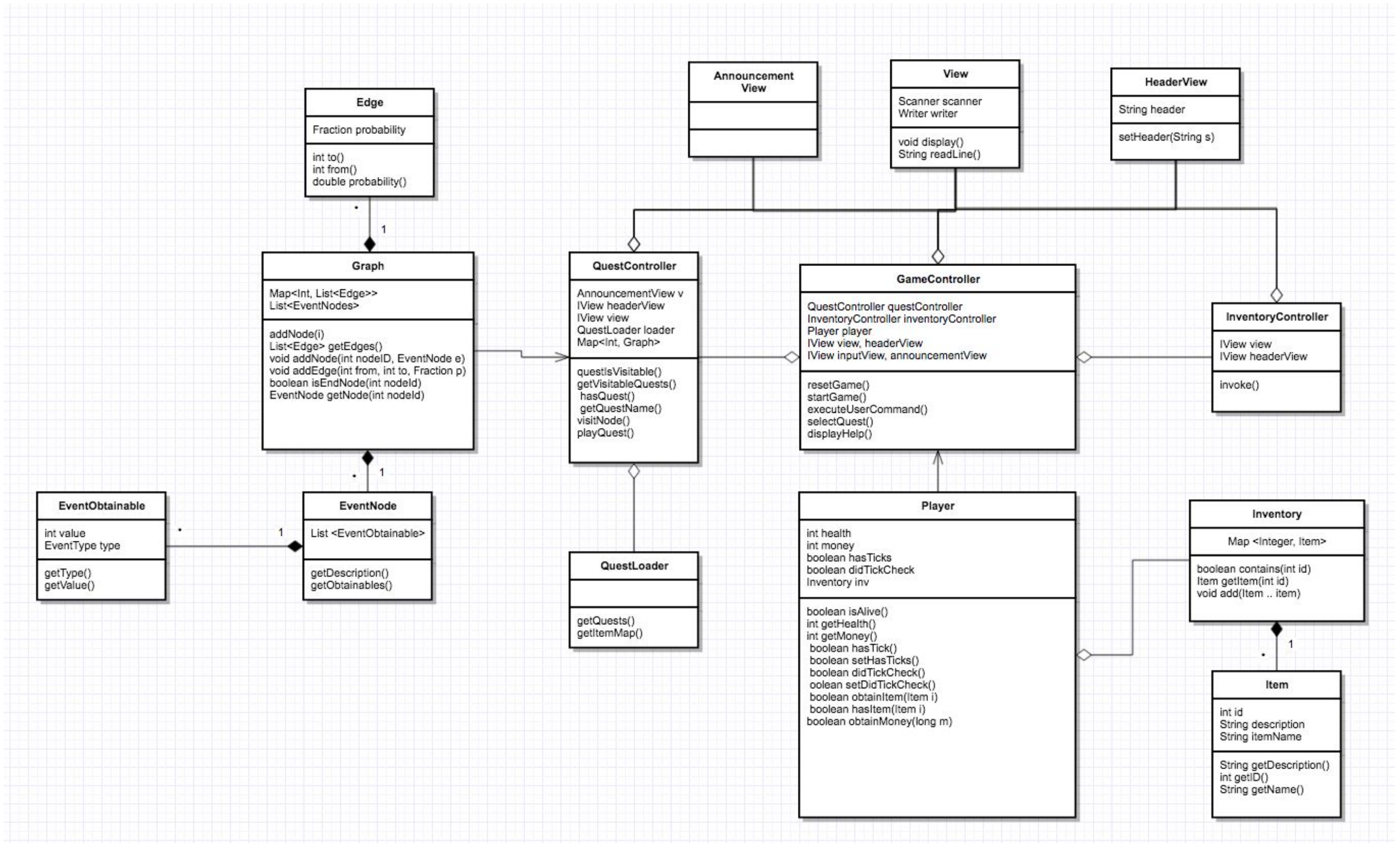
3) Annoucement View: This view is the best view to catch the user's attention. The announcement view turns the text you want to display into uppercase and pads it with fancy ascii art and then displays the result in the console area.



```
~/D/G/TickAttack (master) $ javac *.java; and java GameController

*************************************
****** WELCOME TO TICK ATTACK! ******
*************************************
```

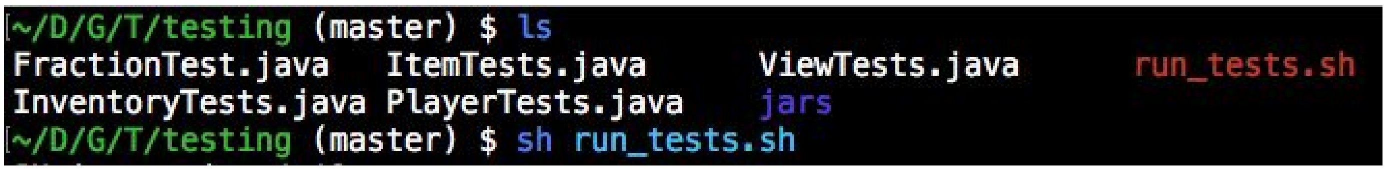
New UML Diagram

You can view our UML diagram in a better resolution by opening the image called "UML.png" at the root of our project folder.



How to run tests

The tests for our program can be found in the testing folder. Inside the testing folder you will find various .java test files, a folder of jars containing hamcrest and junit, but more importantly a shell script called "**run_tests.sh**" which can be used to execute the tests for all the various components of the game:



```
~/D/G/T/testing (master) $ ls
FractionTest.java   ItemTests.java      ViewTests.java      run_tests.sh
InventoryTests.java PlayerTests.java    jars
~/D/G/T/testing (master) $ sh run_tests.sh
```