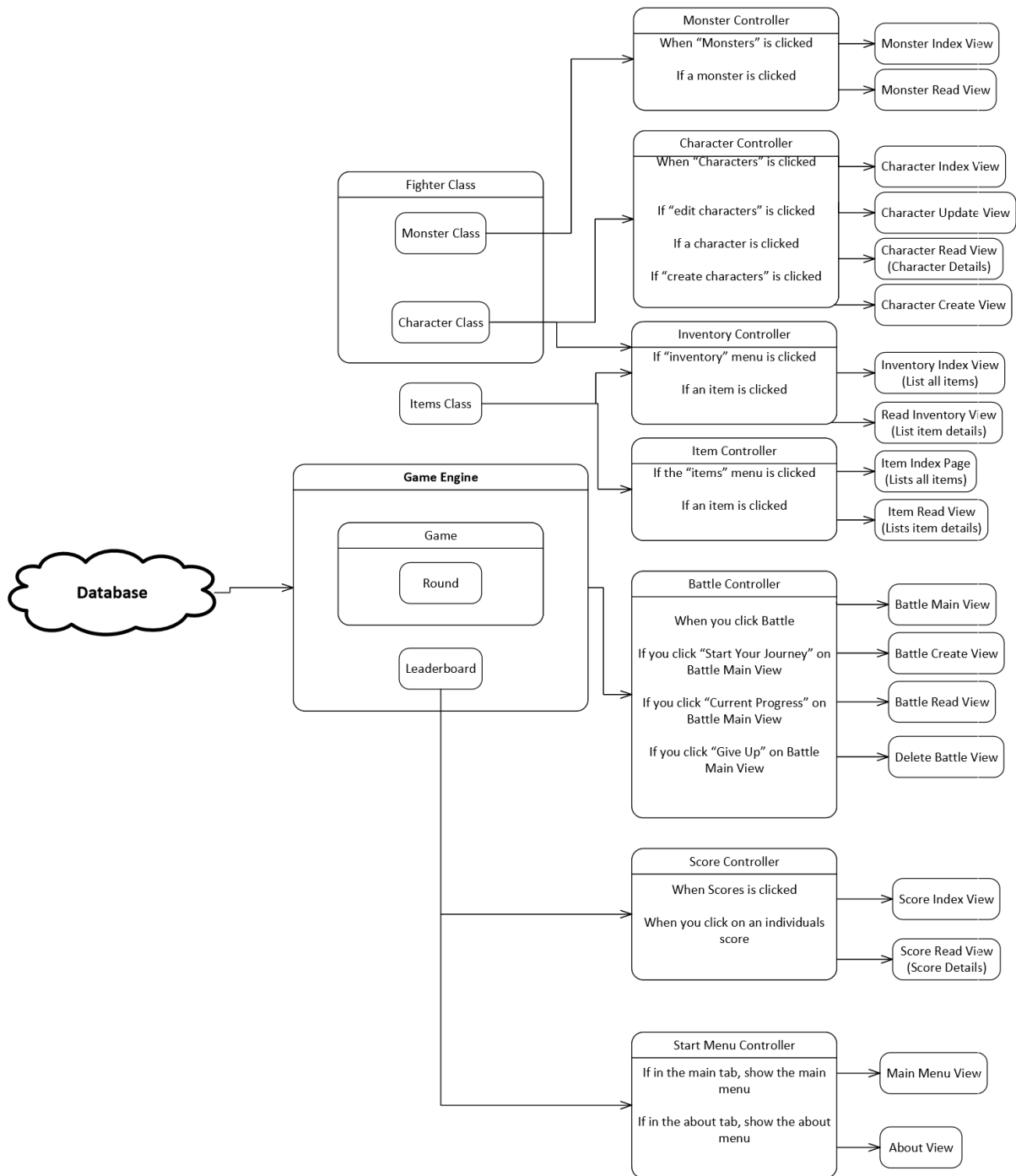
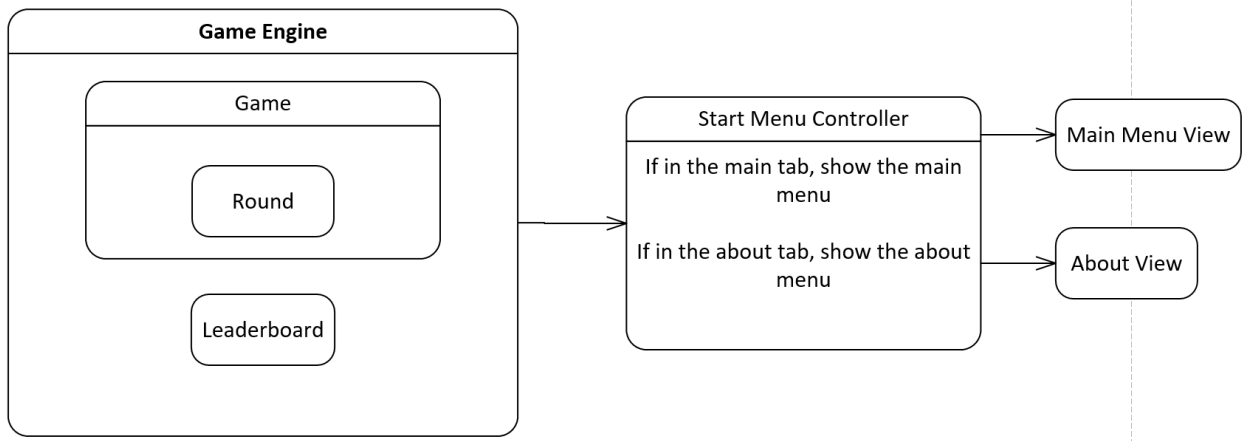


## I. Model-View-Controller Diagram

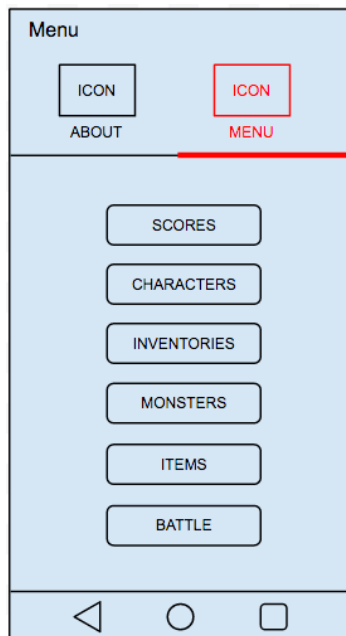


This is broken out with wireframes below.

## II. Screen Wireframes with MVC Integration



**Main Menu View** – occurs in the default tab when the gam launches



- Click on **Scores** to view high scores → jump to Scores Index View
- Click on **Characters** to view character list → jump to Character Index View
- Click on **Inventories** to view items your party has → jump to Inventory Index View
- Click on **Monsters** to view monster list → jump to Inventory Index View
- Click on **Items** to view all possible items → jump to Items Index View
- Click on **Battle** to view battle information → jump to Battle Main View

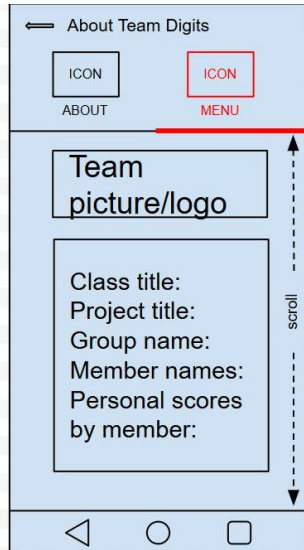
# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### About View

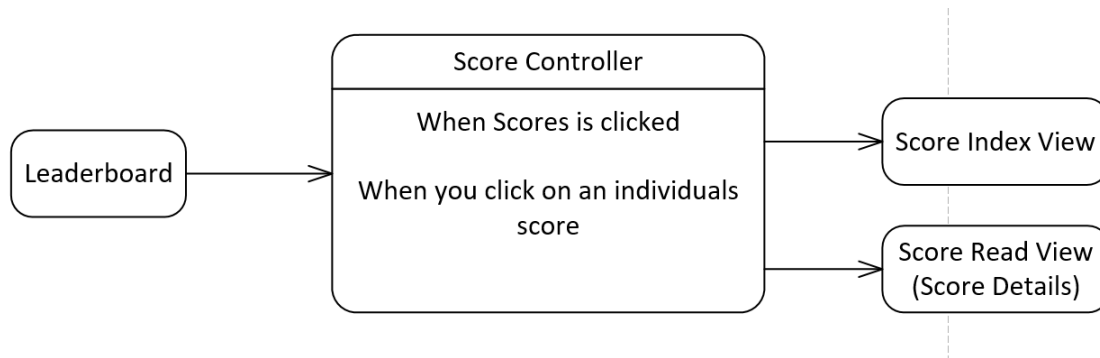
- Occurs anytime you hit the items tab



# CPSC 4910 01 17SQ Mobile App Development

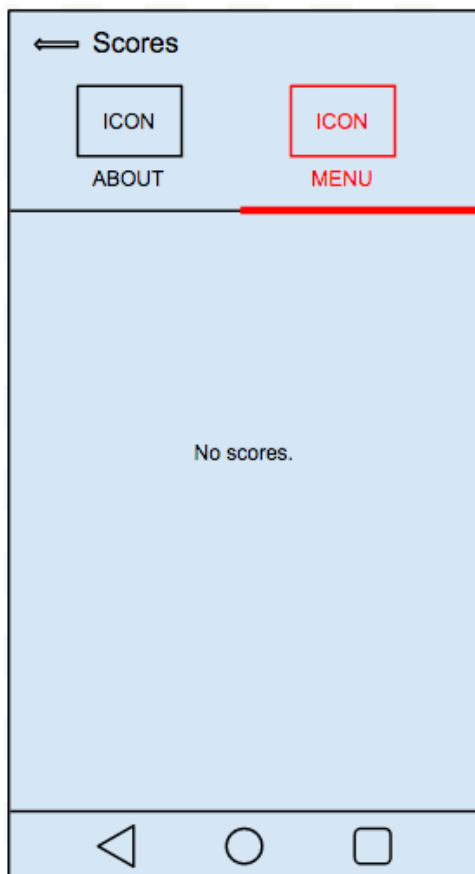
## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

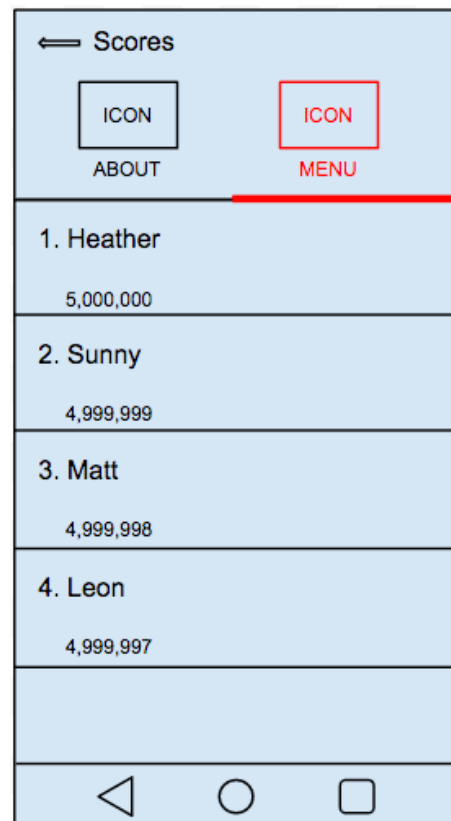


### Score Index View

⇒ *Empty List*



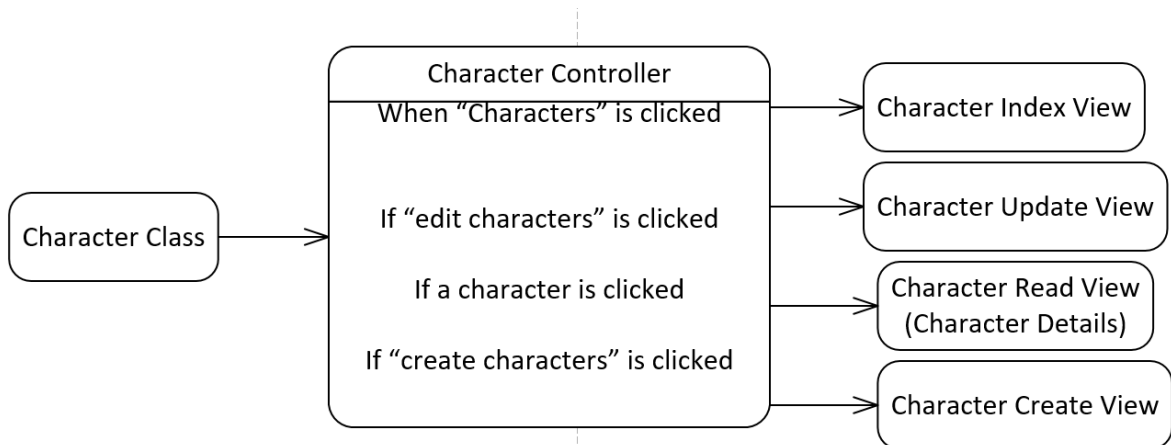
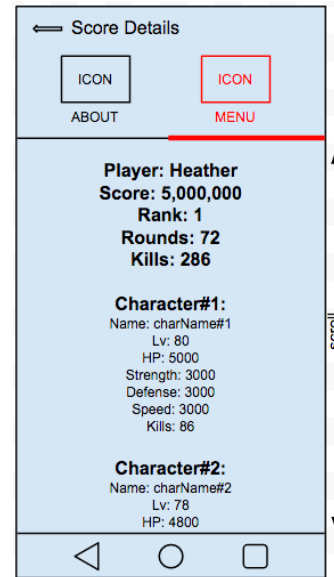
⇒ *List with scores*



- Display a growing list with descending scores
- Click on a **Score Item** to view details → jump to Score Read View

### Score Read View

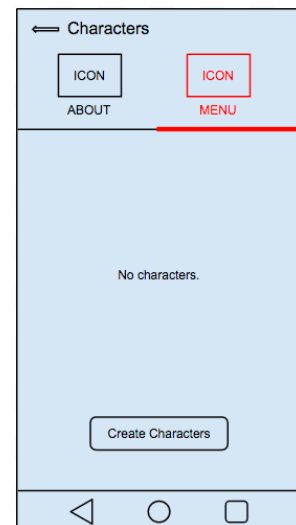
- Shows details of a score



### Character Index View

⇒ *Empty List*

- Click on **Create Characters** to make new characters → jump to Character Create View



# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### ⇒ *List with Characters*

← Characters

ICON ABOUT MENU

CharName#1  
Lv. 1 HP: 15

CharName#2  
Lv. 1 HP: 15

CharName#3  
Lv. 1 HP: 15

CharName#4  
Lv. 1 HP: 15

Edit Characters

- Click on **Edit Characters** to update characters' info. This button is only shown before a battle. → jump to Characters Update View
- Click on a **Character Item** to view details → jump to Character Details View

### Character Create View

← Create Characters

ICON ABOUT MENU

Character 1:  
Name: [sfr]  
Strength: [01]  
Defense: [01]  
Speed: [01]  
Order: [1-4]

Character 2:  
Name: [sfr]  
Strength: [01]  
Defense: [01]  
Speed: [01]  
Order: [1-4]

Character 3:  
Name: [sfr]  
Strength: [01]  
Defense: [01]  
Speed: [01]  
Order: [1-4]

Character 4:  
Name: [sfr]  
Strength: [01]  
Defense: [01]  
Speed: [01]  
Order: [1-4]

Create

### Character Read View

- Shows character details

← Character Details

ICON ABOUT MENU

charName#1

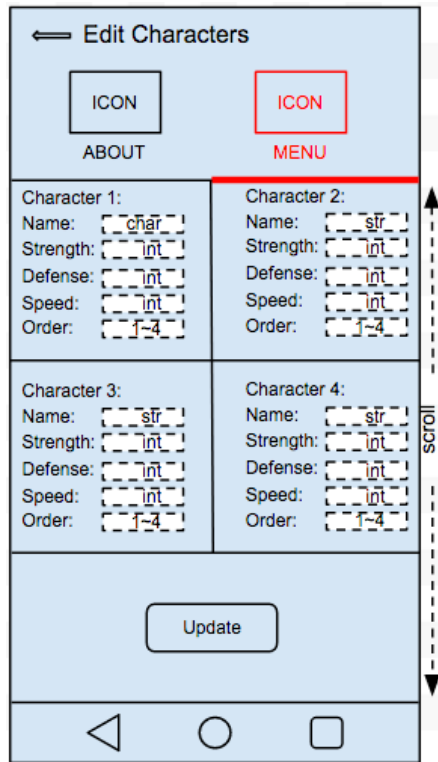
Lv. 1  
HP: 15  
Strength: 12  
Defense: 8  
Speed: 10  
Kills: 0

# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Character Update View



The mockup shows a mobile app screen titled "Edit Characters" with a back arrow. At the top, there are two buttons: "ABOUT" (with a blue icon) and "MENU" (with a red icon). Below these are four character update sections, each with a title (Character 1, 2, 3, 4) and five input fields: Name (text), Strength (int), Defense (int), Speed (int), and Order (dropdown). A vertical dashed line with arrows and the word "scroll" indicates that the character sections can be scrolled. At the bottom is a large "Update" button. The screen is framed by a light blue border, and the bottom shows a standard Android navigation bar with back, home, and recent apps icons.

Edit Characters	
ICON ABOUT	ICON MENU
Character 1: Name: [char] Strength: [int] Defense: [int] Speed: [int] Order: [1-4]	Character 2: Name: [str] Strength: [int] Defense: [int] Speed: [int] Order: [1-4]
Character 3: Name: [str] Strength: [int] Defense: [int] Speed: [int] Order: [1-4]	Character 4: Name: [str] Strength: [int] Defense: [int] Speed: [int] Order: [1-4]
Update	

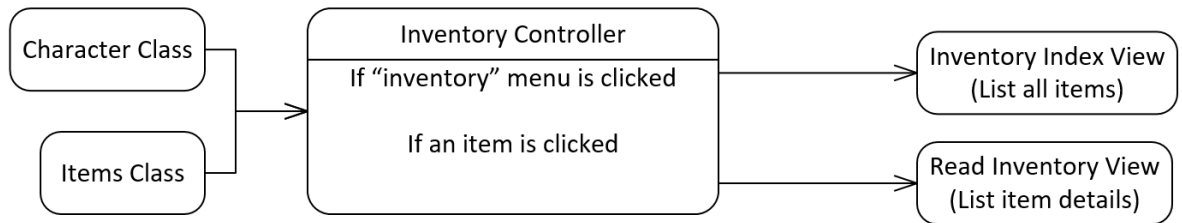
### Notes:

In order to play the game, the user must create four characters. The user is allowed to edit characters in Character Update View, but cannot delete characters.

# CPSC 4910 01 17SQ Mobile App Development

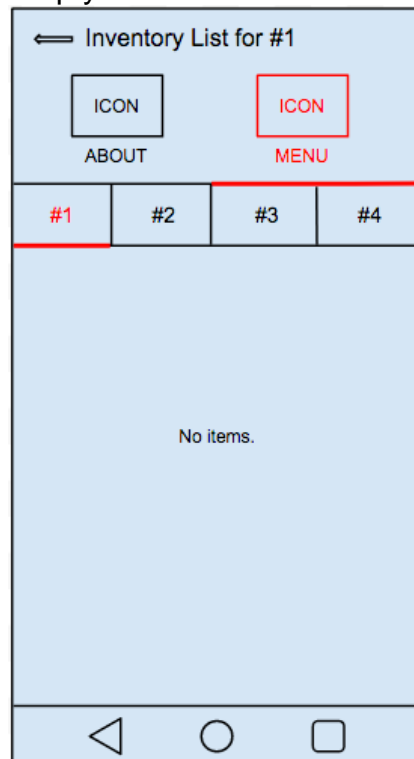
## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

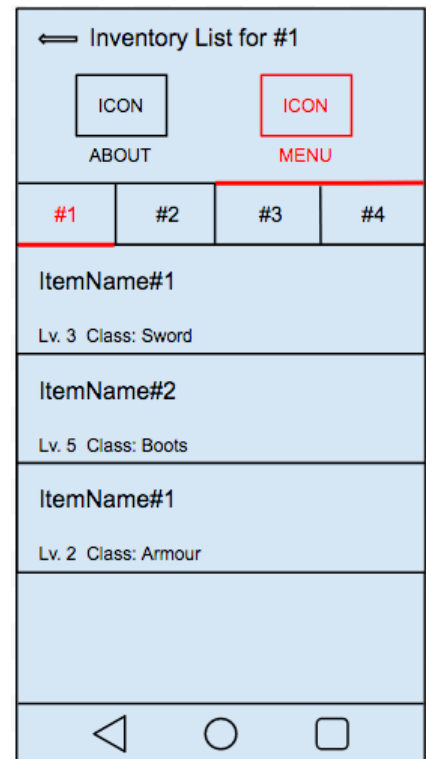


### Inventory Index View

⇒ Empty List



⇒ List with items



- Click on an **Inventory Item** to view details → jump to Read Inventory View



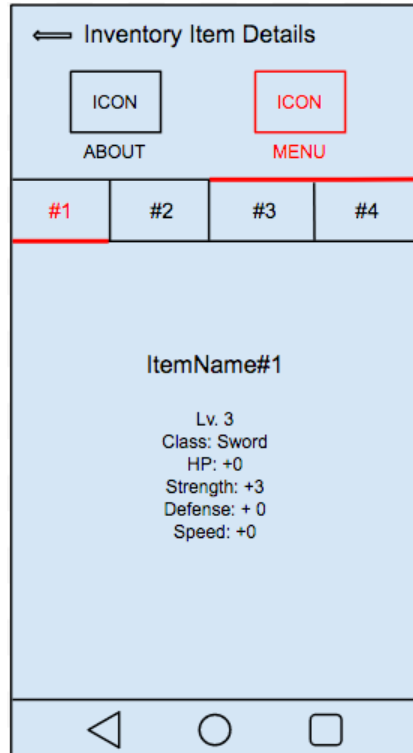
# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Inventory Read View

- Lists all item details

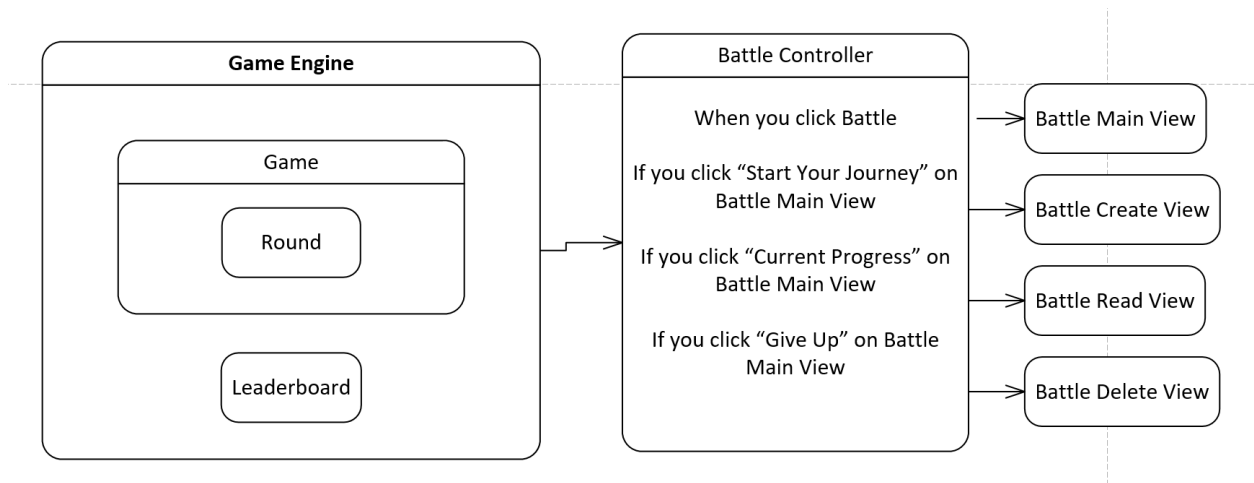


**Note:** Items are randomly generated upon victory against monsters during combat, so no create screen is necessary. Items stack infinitely, so no delete screen is necessary.

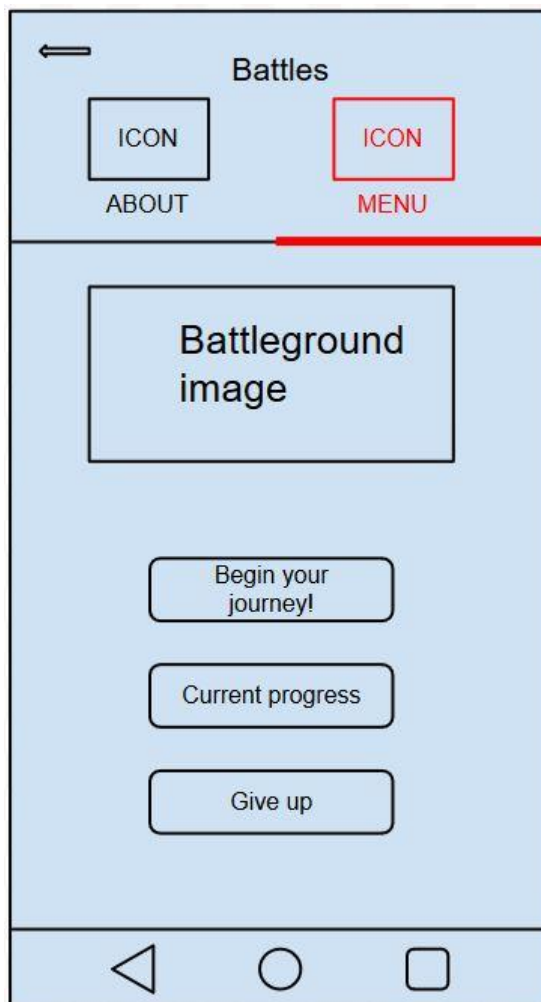
# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)



### Battle Main View



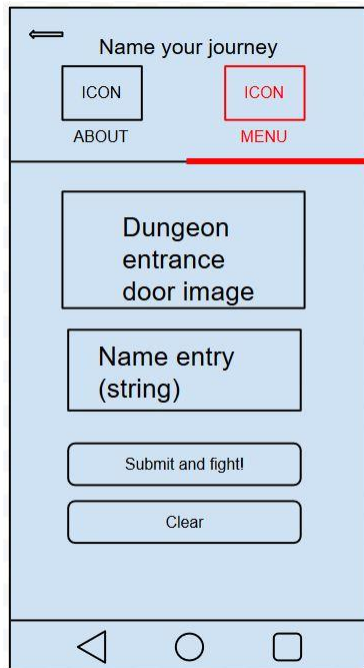
- Anytime you click battle, you end up here first.
- Click on **Start Your Journey** to start a battle → jump to Battle Create View
- Click on **Current Progress** to view the current battle → jump to Battle Read View
- Click on **Give Up** to stop the current game → jump to Battle Delete View

# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

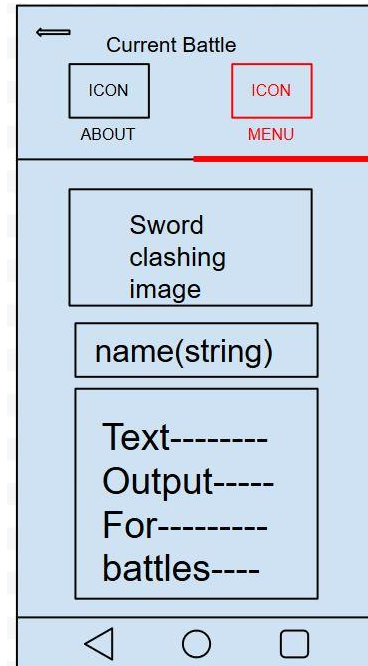
Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Battle Create View



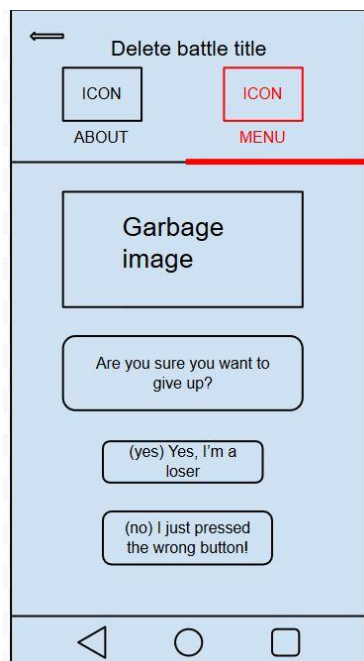
- Enter your name (for high score purposes) and begin the fight!

### Battle Read View



- See all of this game's action

### Battle Delete View

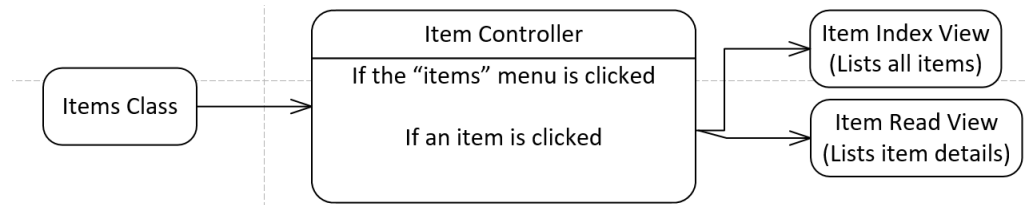


- Stop the current battle

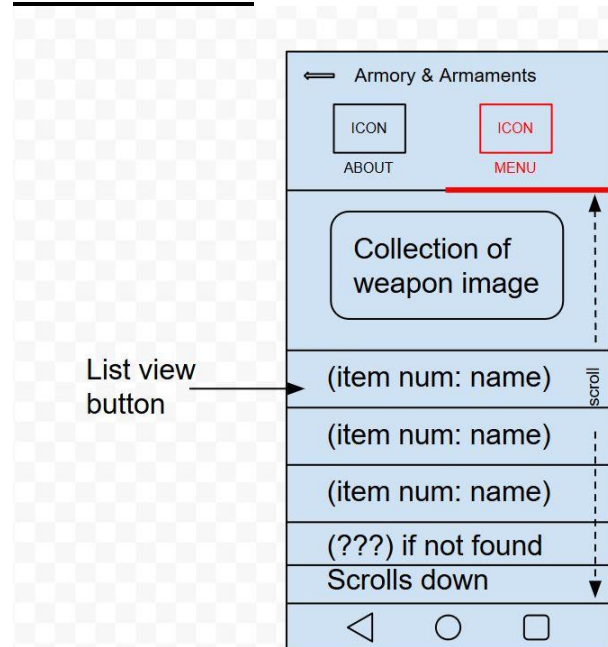
# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

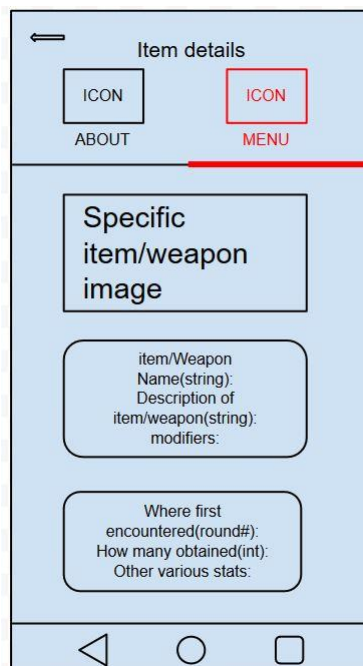


### Item Index View



- Anytime you click Items, you come here.
- Click on an **Item** in the list to view its details -> jump to Item Read View

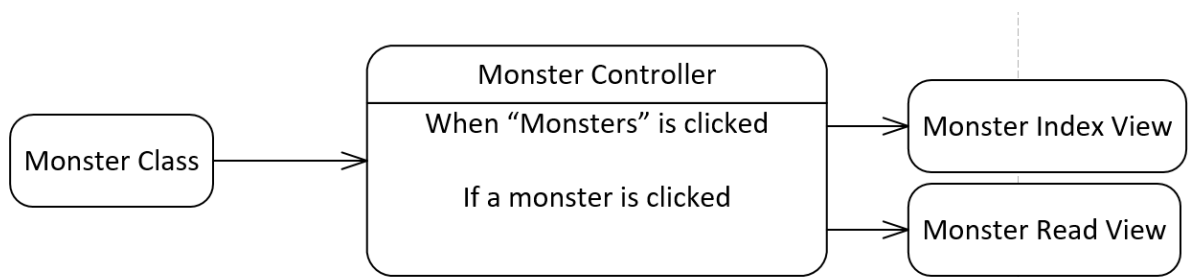
### Item Read View



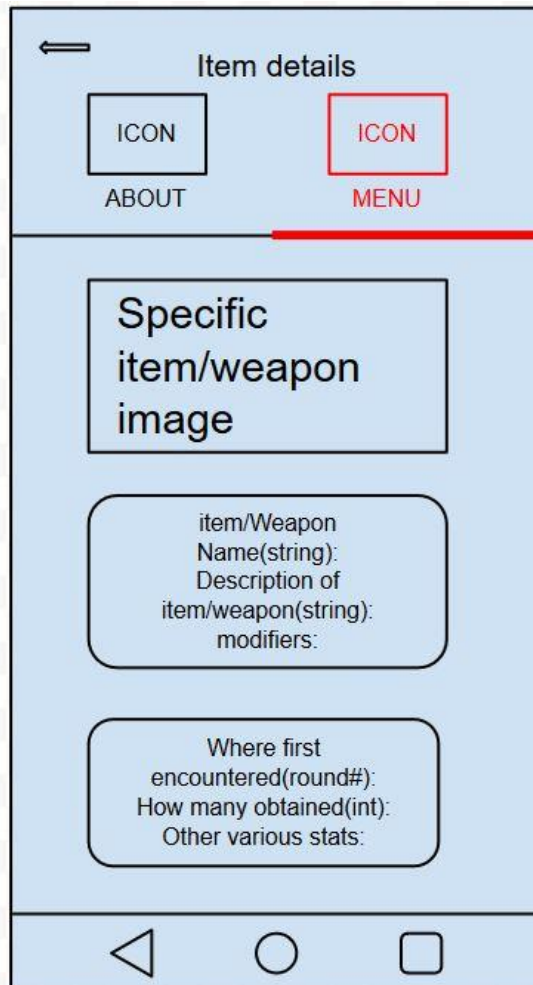
# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)



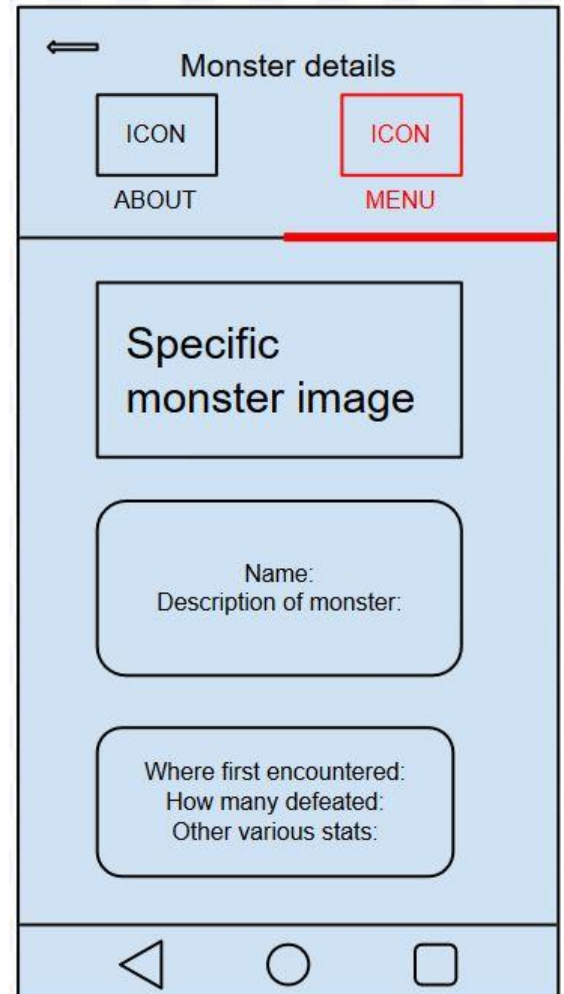
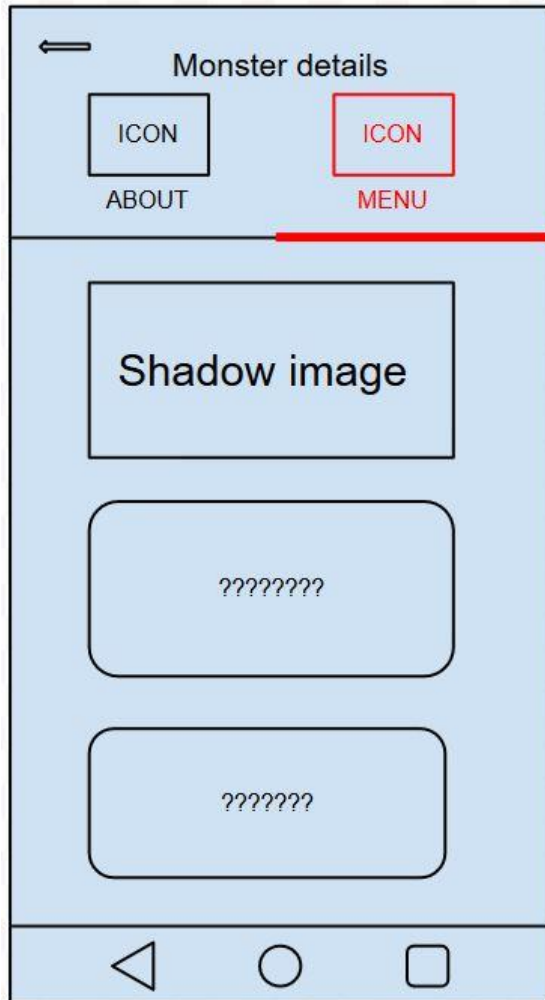
### Monster Index View



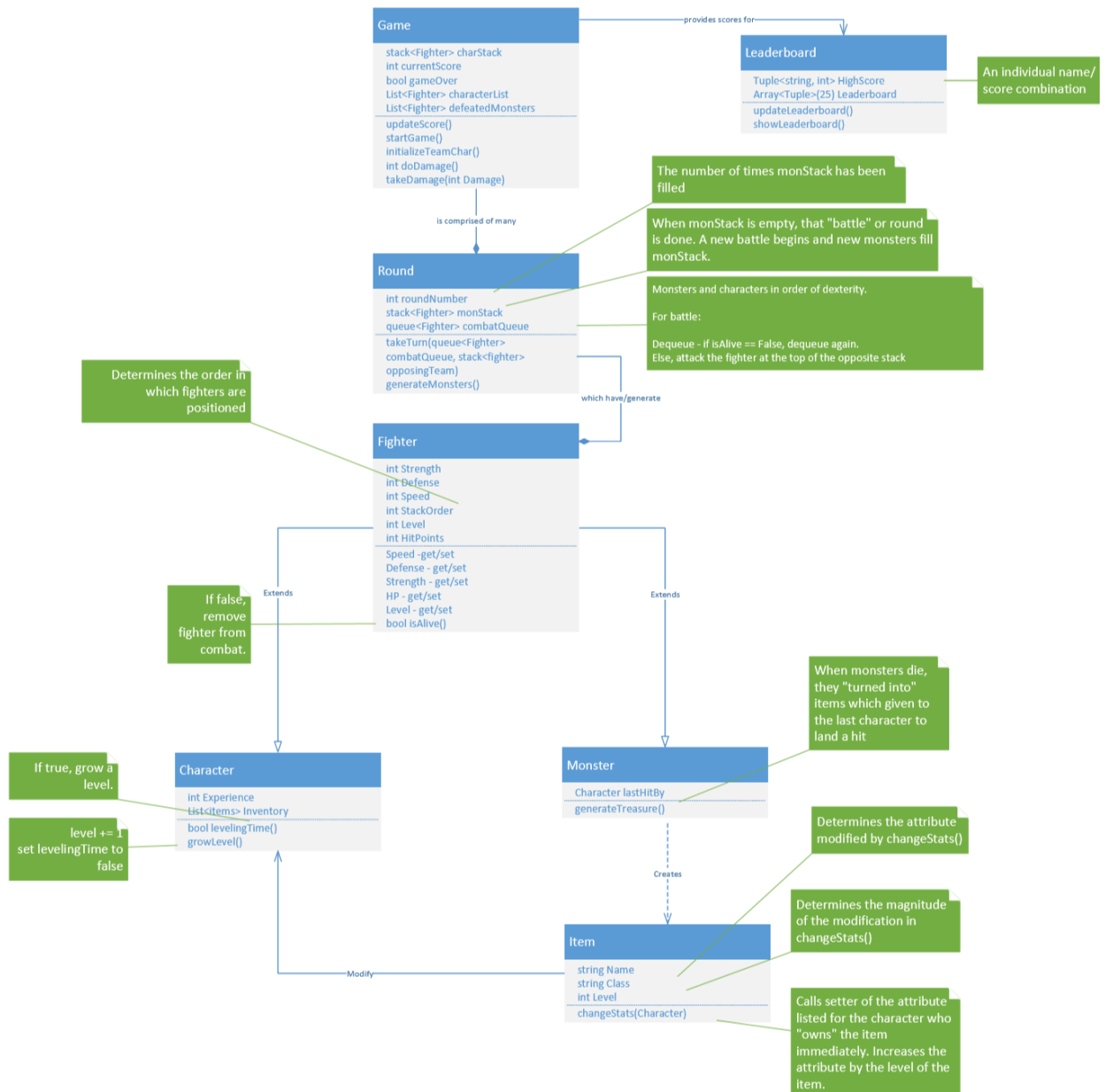
- Anytime you click Monster, you'll come here.
- Click a **Monster** in the list to view its details → jump to Monster Read View.
- ????? are unencountered monsters. When you encounter a monster, their name will replace the ??????????. This idea definitely riff on a Pokédex in Pokémon.

**Monster Read View**

Unseen - Seen



### III. Class Structure for All Components



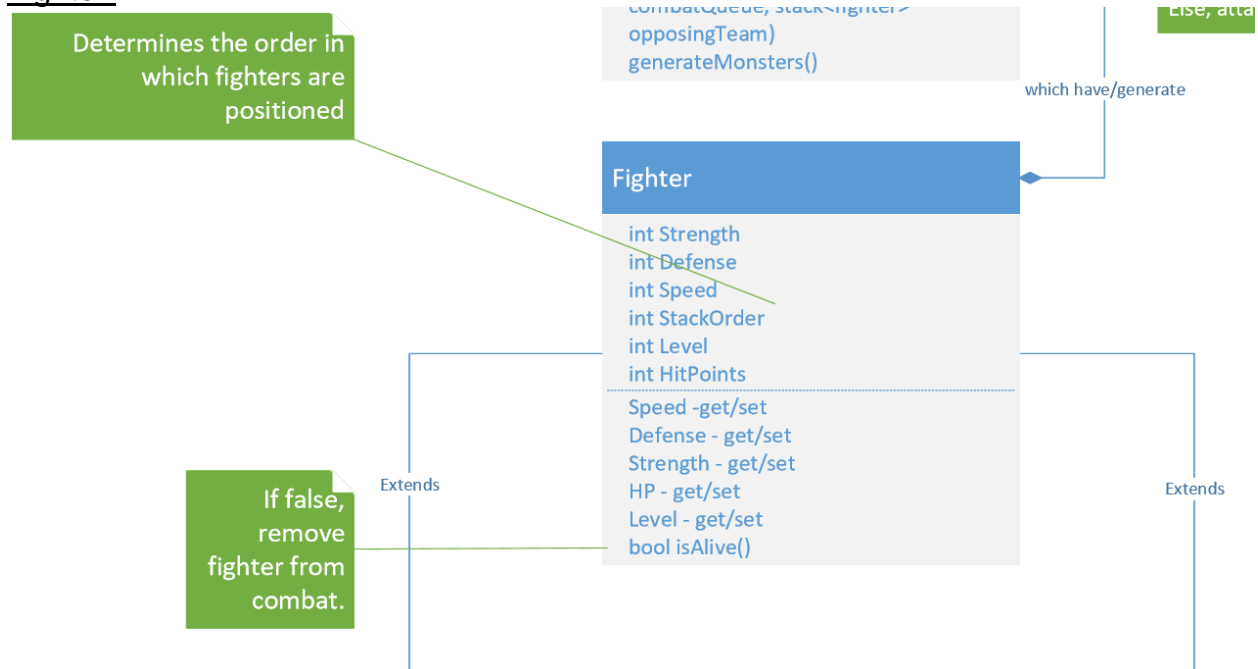
- We have a fighter superclass which contains all the attributes and combat functions shared between characters and monsters. Classes character and monster extend fighter.
- When monsters die, they generate items. Items modify character attributes (explained later).
- There's a game class for individual games (creation, our main while loop, and determining who goes next) and a leaderboard class to store high scores.

# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

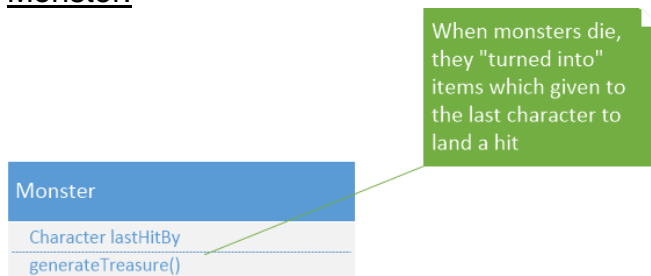
Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Fighter:



The **Fighter** class is the superclass for both **Character** and **Monster**, so it contains the common key attributes shared for both parties in the game. The attributes `Strength`, `Defense` determine the damage done within each turn, `Speed` determines the fight order for each turn (who goes first and second in our turn-based game), and `HP` determines how many damage points that one **Fighter** can take to remain alive. Once a **Fighter** is not alive or defeated, it will be popped out of the battle queue and the next remaining **Fighter** will take over and fight until one party's queue become empty.

### Monster:



The class **Monster** is a child class of **Fighter**. Monsters are stored in a **Monster queue** that is generated by the **Game** class. A new queue is generated each round and when a monster queue is emptied, an item will be generated whose power is based on the total stats of all the monsters defeated. This item will be awarded to the character who delivered the killing blow to the final monster remaining.

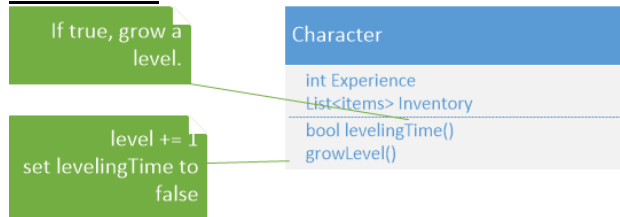


# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

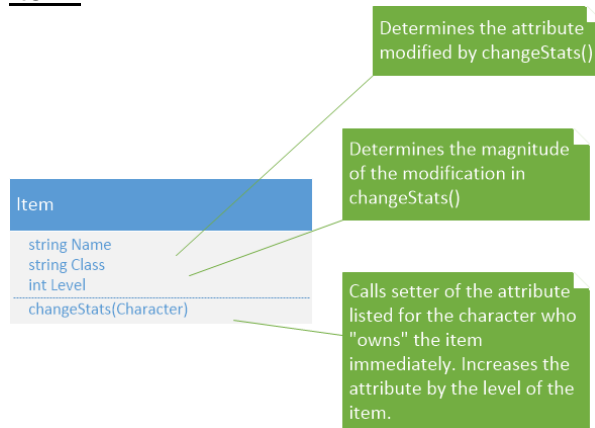
Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Character:



The class **Character** is a child class of **Fighter**. When a **Character** defeats a **Monster**, they will receive an item as a reward. A **Character** can gain experience points and level up so that it will grow stronger as the game proceeds. The **Character** queue will be generated in the **Game** class, and this class will gain and share experiences as rounds are beaten and the team advances.

### Item:



The **Item** dropped by defeated **Monster** will immediately enhance the **Character** that delivered the final blow. The items have different classes, such as armour, weapon, boots etc., as well as the ability to modify the different skills that a fighter has. The number of modification points provided by the weapon will be based on the overall power of the monster team defeated. Armour could potentially enhance the carrier's Defense, a weapon, could increase the carrier's Attack, a pendant could increase the carrier's HPs and so on.

# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Game:

Game
<pre>stack&lt;Fighter&gt; charStack int currentScore bool gameOver List&lt;Fighter&gt; characterList</pre>
<pre>updateScore() startGame() initializeTeamChar() initializeTeamMon()</pre>

The Game class is where the majority of the game logic occurs (obviously). This is where the team of Characters is created, where game state is stored (game active or over, score). The Game class also initializes each round of fighting between the Team Characters and the Team Monsters. The Game class will create a round of fighting, which will cause a team of monsters for the team to fight. The power of the team of monsters will be based on the total stats of the living Characters. If the round is successfully passed by the Characters, their total score for the game is updated.

### Round:

Round	The number of times monStack has been filled
<pre>int roundNumber stack&lt;Fighter&gt; monStack queue&lt;Fighter&gt; combatQueue</pre>	When monStack is empty, that "battle" or round is done. A new battle begins and new monsters fill monStack.
<pre>takeTurn(queue&lt;Fighter&gt; combatQueue, stack&lt;fighter&gt; opposingTeam) generateMonsters()</pre>	Monsters and characters in order of dexterity.  For battle:  Dequeue - if isAlive == False, dequeue again. Else, attack the fighter at the top of the opposite stack

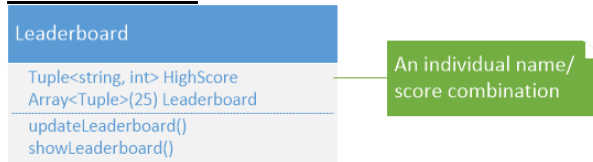
The Round class can have many instances depending on how many rounds the Team Characters can last through the battle against the Team Monsters. In each round, the Character queue and a Monster instance will take turns based on individual speed. Turn order is tracked by a queue. Characters and Monster will be added to a queue, highest speed score first, slowest last and then they will be removed from the queue one at a time to make attacks against the opposing team's tank. Before the Fighter is allowed to attack, it must be verified to be alive. This is where the combat logic for the game will reside.

# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

### Leaderboard:



After a Game reaches the “gameOver” state, that Game instance will send a score to the Leaderboard so that the highest scores can be stored and viewable for the user in a specific screen. The Leaderboard will be updated once a game is done and allow the user to retrieve the score list in descending order from the highest to the lowest for a certain range, say, top 10 or 25.

### III. Game Information:

Our game will run "tank-style." During creation of characters, you can auto-generate characters or allocate skill points yourself (as well as determining the order in a character stack. You can only hit the character at the top of the opposing team’s stack. When they die, they’re popped.

When all of your characters are dead, you’ll be given a score for your party based the amount of experience points gained (as well as some function of the party’s inventory). Therefore, there’s not necessarily a “winner” - just those who have created characters and ordered their battle queues with enough strategy to earn their entire party a place on the leaderboard.

### IV. Combat logic:

When creating a new battle, all remaining characters are enqueued with the fresh batch of newly generated monsters in order of dex. This is the combatQueue, which determines move number.

1. Dequeue the combatQueue. If isAlive == false, dequeue again. Else, attack the fighter at the top of the opposite team’s stack.

Damage done is calculated by:

$$(1-20)(\text{AttackerLevel} + \text{AttackerStrength}) - (1-20)(\text{DefenderLevel} + \text{DefenderDefense})$$

Where (1-20) is a simulated D20 dice roll equalling between 0.1 to 2.0. Negative scores are misses.

3. If there is a hit,  $\text{DefenderHP} -= \text{Damage}$ .
4. Check if the character is alive - `isAlive()`. If it is not alive, dequeue. If the dead fighter is a monster, generate an item and grant it to the last character to land a hit. Update the score. If the fighter is alive, enqueue the fighter in the combatQueue.

# CPSC 4910 01 17SQ Mobile App Development

## Week 3 - Screens UX, Class Design Characters

Submitted by: Heather Wensler, Matthew Irwin, Sunny Yeung, Shuai Miao (Leon)

5. If there are characters remaining, repeat until one stack is empty. If not, go back to 1 and proceed with the next in line to attack.
5. If monster stack is empty but the character queue is not, display that you've entered a new portion of the dungeon. Generate a new round. (roundNumber++, new monsters, new combatQueue). Update the score. Start at 1.
8. When the character stack is empty, update the score and end the game. If applicable, add score (and name) to the leaderboard.

### **V. How Items Work:**

Items are generated by monsters upon their death. They'll be given a name, an attribute they modify, and a level. They are given to the last character to land a hit. This character's base attribute listed they'll immediately increase by the level of the item. Items are only setters. They are stored as a list.