CSC 360 Programming Assignment #1
Due date: Friday, August 29

This assignment will test your ability to write several classes using proper forms of information hiding (visibility modifiers), message passing and arrays of objects. It will also test your ability to read and understand UML notation. Specifically, you will implement an "Adventure Game" by defining a series of classes:

Fighter
Weapon
Armor
Treasure

In addition, you will have a class with a main method that utilizes these classes to play the game. The main method will primarily interact with the Fighter class, and the Fighter class will interact with the other classes. Thus, Fighter will contain most of the functionality for your program.

Main will create an array of Fighters, and the user as a Fighter. Using a for loop, main will iterate through the array of Fighters and the user will fight each one, one at a time. After each fight, assuming the user wins, main will display a menu of options for the user to select the next action. These are:

1. Utilize a magic potion or a scroll (both of which will be among the Treasure)
2. Rest
3. Quit
4. Fight next Fighter

Note: You may add additional choices if you want.

The Fighter class consists of the following instance data: a Weapon, up to two pieces of Armor (one of which will be a shield if available), some Treasure, hit points and a random number Generator. Your constructor will receive all of these except for the second piece of Armor which will initially be null. Aside from accessor methods and a toString, the other methods are described here. A method called fight(Fighter f) will cause *this* Fighter to fight the Fighter f. Each Fighter will take turns. The number of attempts of an attack per turn will be based on *this* Fighter's Weapon which has an attacksPerTurn value. For each attack, generate a random number 1-20. If this number > f's defense (as determined by f's Armor), then you score a hit. The amount of damage is a random number generated between the weapon's minDamage and maxDamage. For instance, the user will initially have a sword with 2 attacks per turn and damage of 1-8 per hit. Both Fighter's (this and f) take turns with the user (this) going first. If f dies after the attack, f does not get its turn, to test to see if f is still alive. The isAlive method is a Boolean method that returns true if f's hit points > 0, false otherwise. If you hit f, use takesDamage to pass the number of hit points of damage to f.

Once the user defeats the Fighter (assuming the user does), the user can take f's Weapon and Armor if 1. The item is a separate item from its body (e.g., if the Fighter has a Weapon of claws or an Armor of hide then the user cannot take it), and 2. It is better than what the user currently has. This requires that you implement a isBetter methods in both Weapon and Armor. For Armor, one piece is better if it has greater protection. A Weapon is better if its average amount of damage is greater than the other. For instance, if one Weapon has 4 attacks per turn with

damage 1-5 and another has 2 attacks at 1-8, then the average is 4 * (1 + 5) / 2 versus 2 * (1 + 8) / 2 or 12 versus 9. Also, if the user defeats f, then the user takes all of f's Treasure. As Treasure will be implemented as an ArrayList, you can use addAll as in

```
this.treasure.addAll(f.getTreasure( ));
```

Now that the user has defeated the Fighter and taken its stuff, display a menu of choices as shown above (1-4). If the user chooses to rest, the user's Fighter will regain a random amount of hit points (11-20) but the user might be attacked by the next Fighter (50% chance) interrupting the rest period and therefore will not reclaim any hit points. Note that the user can only rest once in between Fighters. If the user has any potions or magic scrolls, the user can drink/cast them. The user can use as many as he or she has their Treasure list. After each one is used, remove it from the Treasure's ArrayList. Each potion will heal the user of hit points as follows:

30% chance of healing by 5-10 hit points
40% chance of healing by 11-20 hit points
20% chance of healing by 21-30 hit points
10% chance of hurting by between 1-20 hit points (note that this could kill the user!)

Each scroll will either double the user's hit points (60% chance), do nothing (35% chance) or kill the user (5% chance).

Upon quitting, output your final hit points and the number of Fighters you successfully defeated.

Populate the game as follows:

The user:

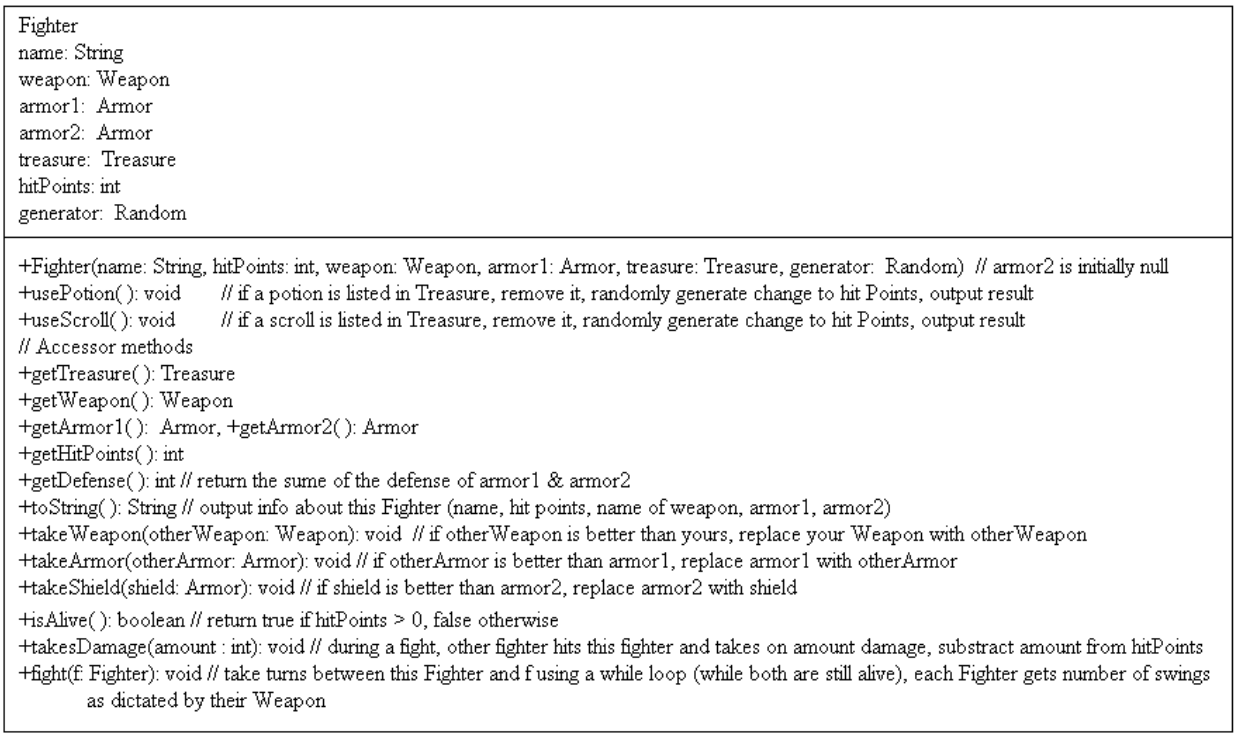
Weapon: Sword, 2 attacks per turn, 1-8 hit points of damage per hit
Armor: leather armor, protection 10
Treasure: 2 potions, 1 scroll
Hit points: 50

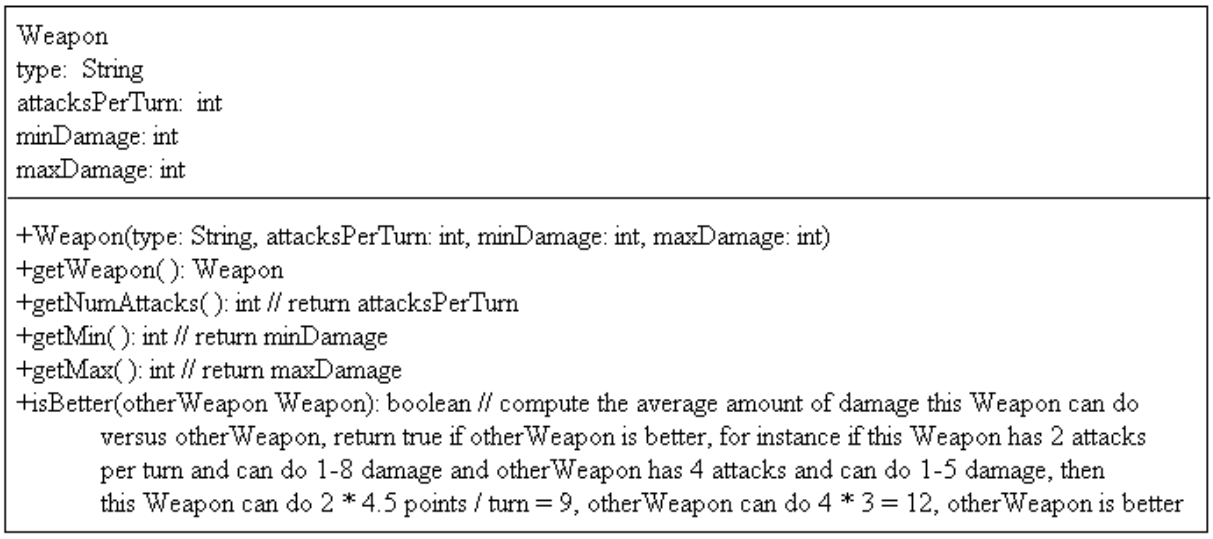
The array of Fighters

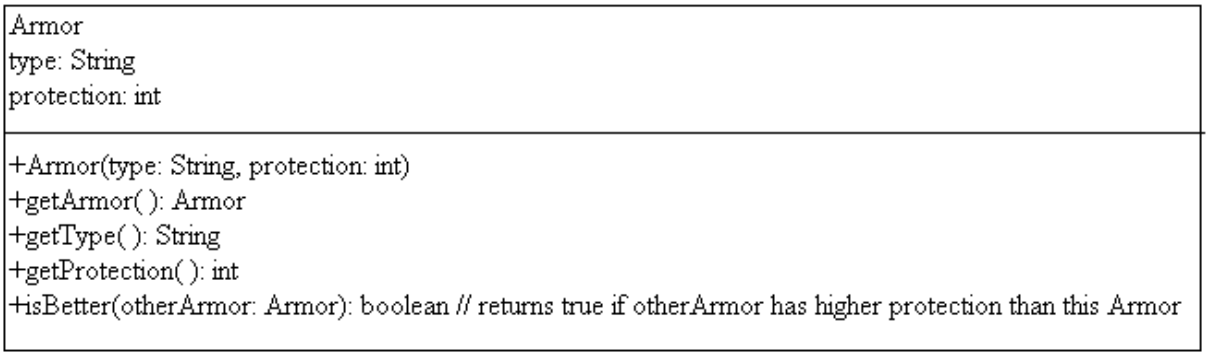
0. Orc (Hit points: 22, Weapon: dagger (3, 1-4), Armor: chain mail (12), Treasure (gold, 1 potion)
1. Nest of snakes (Hit points: 12, Weapon: venom (1, 4-20), armor: none (0), Treasure (1 potion)
2. Troll (Hit points: 35, Weapon: hands (4, 1-8), Armor: hide (14), Treasure (gold, silver, 2 potions)
3. Berserker (Hit points: 28, Weapon: sword (4, 1-4), Armor: Skin (3) & Shield (add 2 to defense), Treasure (silver, 1 potion) (the Shield can be added by the user)
4. Ninja (Hit points: 40, Weapon: super sword (4, 2-8), Armor: chain mail (14), Treasure (2 potions, 1 scroll)
5. Dragon (Hit points: 25, Weapon: claws (2, 11-20, Armor: Skin (14), Treasure: gold, silver, silver, silver, 1 scroll)
6. Doppleganger(Hit points: same as you, Weapon: same as you except only 1 attack per turn, Armor: same as you for main armor but also a Heavy Shield (add 4 to defense), Treasure: same as you)

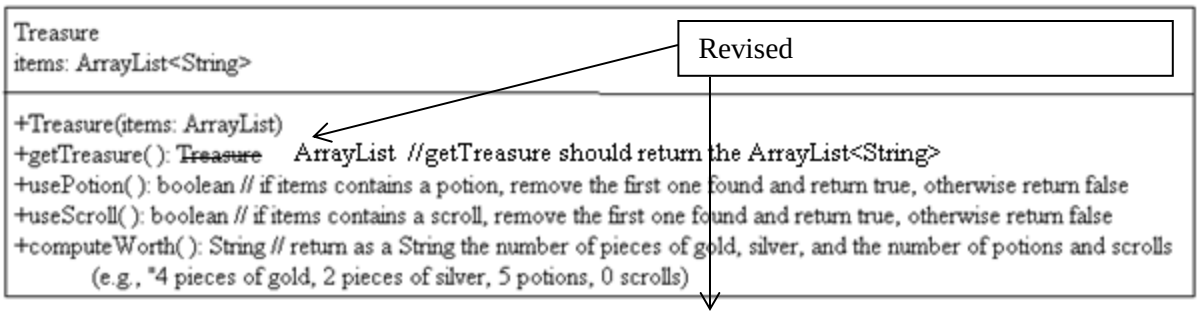
7. Wizard (Hit points: 150, Weapon: lightning bolts (1, 21-40), Armor: magic aura (15), Treasure: gold, gold, gold)

UML for the classes:









NOTE: you cannot construct the Treasure object by passing it a list of Strings as in

```
new Treasure("Gold", "Gold", "Gold); or
new Treasure(new ArrayList<String>("Gold", "Gold", "Gold");
```

Therefore, you will have to create an ArrayList in main before you instantiate your Fighters and add the Strings to the ArrayList, then pass the ArrayList to the Fighter's constructor, clear the ArrayList and add the next Fighter's Treasure to it, etc.

Skeleton of main program:

```
// instantiate a random number generator to pass to each Fighter
Fighter me = new Fighter(…);
Fighter[ ] them = …
int n=0;                    // n is the Fighter who is next up
while(n<8&&me.isAlive( )&&userChoice!=3&&userChoice!=4) {
        me.fight(them[n]);
        if(me.isAlive( )) {
        // display menu and get user input
        // inner while loop
        // do the operation selected, if they choose rest, determine if the next Fighter
        //    appears to fight early, then exit inner loop
        //    else if choice is 3 or 4, exit inner loop
        //  display menu and get user input
        n++;
}
If userChoice is 3, indicate that the user quit early
Otherwise if me.isAlive( ) output that the user wins and output the user's final Treasure
Otherwise output that the user lost and how many Fighters the user killed before dying
```