COMP 202 FINAL EXAM LONG ANSWER PRACTICE

CARLOS G. OLIVER

1. Q1: Apocalypse Simulator

1.1. **Background.** We are going to create a little game that simulates a world where a zombie outbreak is slowly taking over the human population. The simulation will take place on a square grid which will define our world. We initialize our world to have some zombies and some humans randomly scattered on the grid. During the simulation, zombies and humans will be moving around this grid. If a zombie and a human occupy the same position at the same time, they will have to fight each other. If a human is killed by a zombie, the human becomes a zombie and if a zombie is killed by a human the zombie just dies.

1.2. **Programming.** To do this, you're going to create two classes: `World`, `Survivor`. The `Survivor` class is going to define a person in this world. Survivors can either be infected and they act like zombies, or they are not infected and they act like humans. They will also have a health, x and y position and the ability to attack, so we will have the following attributes and methods:

- `infected`: a `boolean` that is `true` if the survivor is infected and false otherwise.
- `health`: `int` whose initial value is 100.
- `xPos`: `int` storing the x position of the survivor.
- `yPos`: same but for y
- `strength`: `double` between 0 and 1 that defines the probability a survivor's attack will be successful.
- `attack(Survivor s)`: non-static method that reduces the health of `s` by 10 with probability `strength`. If the victim is not infected and their health reaches zero or less this method sets the victim to infected and returns their health to 100. If the victim is infected and their health reaches zero or less nothing happens.
- `getPosition()`: non-static method that returns the x and y position of the survivor as an array of size 2 where x is the first index and y is the second.
- `get` and `set` methods for the following attributes: `health, infected, xPos, yPos` (i know this is tedious, i'm sorry)

Of course, you are going to need to create a constructor for the `Survivor` class. The constructor header will take as input values for the following attribues (in this order): `infected`, `xPos`, `yPos`, `strength`.

Date: June 15, 2017.

The `World` class will initialize all the survivors, move the simulation forward in time and print a visualization of the world. More specifically, the `World` consists of an array of `Survivors`, and some methods that update this array and display it for the user.

The `World` constructor will take as input three integers: (i) the number of healthy survivors to start with, (ii) the number of infected survivors to start with (iii) the size of the world, i.e. a world of size 5 will be a 5x5 grid. The constructor will then initialize an array of `Survivors` of the proper size and store the newly created `Survivor` objects by repeatedly calling the `Survivor()` constructor with random values for position and strength. Make sure you create the right number of humans and zombies at this point. The `World` object will then have the following attributes and methods:

- `survivors`: array storing all the survivors.
- `numHumans`: static field that stores the number of non-infected survivors.
- `numZombies`: static field that stores the number of infected survivors.
- `updateWorld()`: non-static method that updates the state of the world by doing the following:
  1. If any survivor has the same x and y position as another, the survivor attacks the other survivor. Make sure here a survivor doesn't attack themselves, and a surivor only attacks a survivor of the opposite kind (i.e. zombies don't attack zombies and humans don't attack humans). If a zombie or human is killed update `numZombies` and `numHumans`.
  2. Each survivor then moves one step randomly: (i) randomly pick the axis of the move (x or y) (ii) randomly pick the direction along that axis (iii) move one space in that direction. If the move would take the survivor outside the bounds of the grid, the player appears on the opposite side of the grid. e.g. grid size is 5x5 and survivor is currently at `(2, 4)`. Increasing y by 1 would result in the new position being `(2, 0)`. This is known as periodic boundary conditions.
- `toString()` override the default `toString()` method to print the world nicely. For each position in the grid you will find all the survivors that occupy it (assume you have a helper method `getSurvivors(int x, int y)` which returns an `ArrayList` containing all survivors at position `x` and `y`. If no survivor occupies that space, print `"-------"` A space occupied by survivors with health greater than zero prints a string like this `"HZH"` which would be two humans and a zombie in the same space.

Finally, our main method (in the `World` class) will simply ask the user for a world size, number of humans, number of zombies, and a number of steps using `Scanner`. Input with improper format should be handled with a try/catch (no need to ask again for input, the program can exit but with a message saying input format was incorrect.) It will then (i) create the `World` (ii) for the number of steps given repeatedly call `updateWorld()` and print the world.