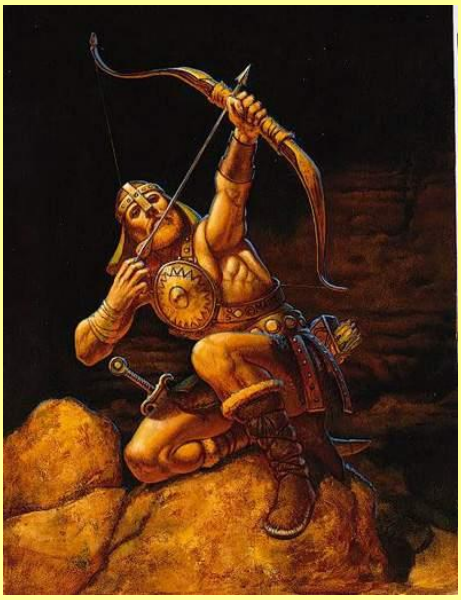
Assignment - Inheritance and Polymorphism
75 points
(See Canvas for due date)

Heroes vs. Monsters



For this assignment you will create an inheritance hierarchy and utilize polymorphism. The idea behind the game is that one of three Heroes (Warrior, Sorceress, Thief) chosen by the user will battle one of three randomly selected Monsters (Ogre, Gremlin, Skeleton). The battle is to the death. When a character dies, the user can play again.

The inheritance hierarchy for the program is as follows:

DungeonCharacter (base - abstract)

- contains instance variables that any character in the game will have -- protected access is ok (NO public access allowed!)
  - name - **String**
  - hit points (how much damage a character can take before it expires) - **integer**
  - attack speed - **integer** (1 is slowest)
  - damage range (minimum and maximum amount of damage a character can do on an attack) - **two integers**
  - chance to hit opponent when attacking - **double**
  - anything else you deem necessary
- constructor to initialize instance variables

- get and set methods as you deem necessary
- an **attack** method
    - first checks if character can attack (based on chance to hit)
    - if it can, a hit in range of minimum to maximum damage is generated and applied to opponent -- user should be informed of what happens
    - if it cannot, a message should be displayed that says the attack failed

Hero (derived from DungeonCharacter -- Hero is also abstract)

- contains instance variables that any hero should have
    - chance to block (a Hero has a chance to block any attack by a Monster -- this should be checked after the Monster has attacked and hit points are about to be removed from the Hero) - **double**
    - number of turns in round (number of attacks/special operations a Hero gets to perform per round -- based on Hero attack speed versus Monster attack speed - if Hero is twice as fast then s/he gets two turns, three times as fast gets three turns, etc.) - **integer**
    - anything else you deem necessary
- constructor - should call base constructor, too
- a method to get name of character from user -- should be invoked in constructor after base constructor
- get, set, and any other methods (this includes overridden ones) you deem necessary

Warrior (derived from Hero)

- instance variables as you deem necessary (none may be necessary!)
- a **crushingBlow** method that does 75 to 175 points of damage but only has a 40% chance of succeeding -- this is the Warrior's special skill
- gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Warrior)
- suggested statistics for Warrior (should be set up in constructor(s))
    - hit points: 125
    - attack speed: 4
    - chance to hit: 0.8 (80 percent)
    - minimum damage: 35
    - maximum damage: 60
    - chance to block: 0.2 (20 percent)

Sorceress (derived from Hero)

- same game as Warrior except special skill is **heal** (choose a range of hit points that will be healed)
- suggested statistics for Sorceress (should be set up in constructor(s))
    - hit points: 75
    - attack speed: 5
    - chance to hit: 0.7 (70 percent)
    - minimum damage: 25
    - maximum damage: 45
    - chance to block: 0.3 (30 percent)

Thief (derived from Hero)

- same game as Warrior except special skill is **surpriseAttack** -- 40 percent chance it is successful. If it is successful, Thief gets an attack and another turn in the current round. There is

a 20 percent chance the Thief is caught in which case no attack is rendered. The other 40 percent is just a normal attack.

- suggested statistics for Thief (should be set up in constructor(s))
  - hit points: 75
  - attack speed: 6
  - chance to hit: 0.8 (80 percent)
  - minimum damage: 20
  - maximum damage: 40
  - chance to block: 0.4 (40 percent)

Monster (derived from DungeonCharacter -- abstract)

- contains instance variables that any Monster should have
  - chance to heal (a Monster has a chance to heal after any attack that causes a loss of hit points -- this should be checked after the Monster has been attacked and hit points have been lost -- note that if the hit points lost kill the Monster, it cannot heal itself!) - **double**
  - minimum and maximum points a monster can heal itself - **two integers**
  - anything else you deem necessary
- constructor - should call base constructor, too
- get, set, and any other methods (this includes overridden ones) you deem necessary
- a **heal** method that is based on chance to heal and then range of heal points for monster

Ogre (derived from Monster)

- instance variables as you deem necessary (none may be necessary!)
- gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Ogre)
- suggested statistics for Ogre (should be set up in constructor(s) -- choose a name for your Ogre)
  - hit points: 200
  - attack speed: 2
  - chance to hit: 0.6 (60 percent)
  - minimum damage: 30
  - maximum damage: 60
  - chance to heal: 0.1 (10 percent)
  - minimum heal points: 30
  - maximum heal points: 60

Gremlin (derived from Monster)

- instance variables as you deem necessary (none may be necessary!)
- gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Gremlin)
- suggested statistics for Gremlin (should be set up in constructor(s)-- choose a name for your Gremlin)
  - hit points: 70
  - attack speed: 5
  - chance to hit: 0.8 (80 percent)
  - minimum damage: 15
  - maximum damage: 30
  - chance to heal: 0.4 (40 percent)
  - minimum heal points: 20
  - maximum heal points: 40

Skeleton (derived from Monster)

- instance variables as you deem necessary (none may be necessary!)
- gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Skeleton)
- suggested statistics for Skeleton (should be set up in constructor(s)-- choose a name for your Skeleton)
    - hit points: 100
    - attack speed: 3
    - chance to hit: 0.8 (80 percent)
    - minimum damage: 30
    - maximum damage: 50
    - chance to heal: 0.3 (30 percent)
    - minimum heal points: 30
    - maximum heal points: 50

HeroesVersusMonsters

- contains main method
- contains helper methods to run the game (see Game Play section below)
- should contain a Hero reference and a Monster reference (for playing the game)

Game Play

1. User chooses a Hero
2. Program randomly selects a Monster
3. Hero and Monster battle until one is defeated
    1. Hero goes first. Hero may have multiple attacks depending on speed of Hero versus Monster
    2. Monster goes second and always attacks. Monster cannot attack if Hero defeated it during Hero attack (duh)
    3. After each round user should be given option to quit.
4. After battle is complete, user should be given option to play a new game

Throughout game play the user should be given as much information as possible about the battle. Here are things the user should be told:

- What happened on an attack (how much damage or that it failed or that it was blocked)
- How many hit points remain for a character after the character loses hit points (via an attack) or gains hit points (via healing)
- Anything else you can think of to make game play more clear

---

Group Work

You may work in pairs on this assignment. If you choose to do this, you must complete a write-up that describes very specifically what each person did. You must also implement the extra credit (features 1 and 2 below) as part of the regular assignment. Be sure to insert comments in your submission informing the grader.

Extra Credit if working individually (required if working as a team)

1. 5 points - add a special skill to each Monster. When a Monster attacks this special skill should be randomly chosen (perhaps 30 percent of the time)
2. 5 points - add another Hero. The Hero should have a unique special skill

---

To turn in

Submit a zip file to Canvas Assignments (just one if you work in a team) that contains:

- all source files necessary to compile and run program
- Other documentation (required):
    - anything you could not get to work
    - extra credit attempted
    - your name(s)
    - anything borrowed from elsewhere

Get started ASAP!

NOTE: A prudent strategy is to first write the DungeonCharacter class and make it so two objects of this class can fight (you must temporarily suspend the 'abstract' designation for this class.) Once this works, make DungeonCharacter abstract, and move on to Hero and Monster and do the same. Finally, add the sub-classes of Hero and Monster and test those.