Sir Jav-a-lot must now travel through JavaBean Forest in order to get to his final destination.  Unfortunately for Sir Jav-A-Lot, it’s midnight, visibility is zero and JavaBean Forest is full of the most evil characters!  Not just other Knights, but also Oracle Ogres, Linux Leprechauns and those midnight Wally Wal-Mart customers!    
  
So the idea is that Sir Jav-A-Lot will encounter random threats throughout this journey of which will be generated at run time, hence polymorphism.     
     
1.    Create an Enemy abstract superclass that fights with the Knight with the abstract method fight that must be overridden and returns an int for damage.   
  
2.    The enemy superclass should have a takeDamage method with an implementation in the superclass and maintain an int that represents the enemy’s health.  It should also have all of the appropriate accessors and mutators.   
  
3.    The superclass must also have a static getRandomEnemy method that returns a random Enemy object (use the Random generator in the java library) which it instantiates from one of the three implementations below (this is called a static method factory and is used extensively in the java library).    
  
4.    You must also create a custom Exception method that is thrown by the superclass when negative damage is passed to the takeDamage method as this is invalid (for example InvalidDamageException).   
  
5.    When you set up the attack, request how many enemies your knight sees. Call the getRandomEnemy method that number of times and see how your knight does.  Since you don’t know which Enemy object will be returned you must build the class around Enemy (also called building to an interface (or abstract class) rather than an implementation) . Note that this uses polymorphism as the driver program has no knowledge of the underlying implementation of Enemy.  
  
**Note:  This is a tough concept to grasp, so start early and post ALL questions to the discussion board!**  
  
**Acceptance Criteria:**  
  
1.    Have at least 3 implementations (i.e. ogre, sorcerer, troll, or whatever you can dream up) of the superclass enemy, but only encounter 1 of those implementations randomly. You must override both the toString method inherited from Object defining the enemy and the fight method inherited from the abstract superclass.  
  
2.    Creation of at least one standard exception used in the book.  
  
3.    Creation of at least one custom exception of your own creation.  
  
**Extra Credit:**  
  
1.    Feel free to spice up the fight with different types of moves and attacks that make sense with the type of enemy (like a sorcerer is probably resistant to magic attack and probably uses them often). +10pts  
  
2.    Have at least 3 implementations (i.e. ogre, sorcerer, troll, or whatever you can dream up) of the superclass enemy, with the random encounter of at least 2 of them.  You must override both the toString method inherited from Object defining the enemy and the fight method inherited from the abstract superclass. +10pts  
Zip up the files and title the zip with the following format:  lastname, first initial, underscore, assign5 (ie John Doe would submit DoeJ\_assign5.zip)