COMP 132: Advanced Programming
Spring 2016

Quiz 3, March 1, 2016

In this quiz, you are asked to build a Java type hierarchy representing different kinds of items that can be used in a simple computer game.

In the given Java project, a main program code in Test.java file is provided to test your code. You are going to write code for the following. Decide whether each should be an abstract class, a concrete class or an interface.

- **Item:** Refers to general entity of the game.
- **Armor:** A kind of item that protects the player.
- **Weapon:** A kind of item that the player uses to hit the enemies.
    - **Sword:** A kind of weapon that hits an enemy by cutting.
    - **Bow:** A kind of weapon that hits an enemy by throwing arrows to the enemy.
- **Upgradable:** Bow and Armor objects are upgradable objects. They should implement their own **void upgrade()** method.

Important Note: Declare all of your class fields as private for the sake of information hiding.

The fields and methods of the types above are described below:

- (15 pts) An **Item** has the following members:
    - A **static** field **itemQuantity** of type int with initial value of zero.
      Holds the total number of player items generated so far.
    - A field **itemPrice** of type **double**
    - A field **itemWeight** of type **double**
    - A constructor
      Item(double price, double weight)
        - initializes the **itemPrice** and **itemWeight** fields with the given parameters
        - increments the **itemQuantity** field by one

      Make sure you check the input values properly. The **price** and **weight** cannot be less than zero. Assign them to zero if they are less than zero.
    - A **static** method int **getItemQuantity()** that returns the field **itemQuantity**
    - **Getter** methods for the other non-static fields
    - An **abstract** method **boolean useItem()** that is computed differently for different kinds of items.

- (25 pts) All **Armor** objects have:
    - A field **defencePoints** of type **double**
    - A field **isWorn** of type **boolean**
    - A constructor
      Armor(double price, double weight, double defencePoints)
        - Initializes the object fields with the given parameters
        - Initializes the **isWorn** field to be **false**

      Make sure you check the input values properly. The **defencePoints** cannot be less than zero. If it is less than zero, assign it to zero.
    - **Getter** methods for the **defencePoints** and **isWorn** fields
    - The method **boolean useItem()** sets the **isWorn** field to **true** and returns **true**.

- A **toString()** method that returns the information about the Armor **price**, **weight** and **defencePoints**.
- A method **double getHit(double damagePoints)**: When the armor gets hit, it protects the player by spending **defencePoints**. This method decreases the **defencePoints** and returns the amount of points not protected by the armor.
  - If the armor is worn
    - If the **defencePoints** is greater than or equal to the **damagePoints**, it decreases **defencePoints** by **damagePoints** and returns 0.
    - If the **defencePoints** is less than the **damagePoints**, the method sets **defencePoints** to 0 and returns the remaining amount of **damagePoints** (i.e. the **damagePoints** minus the **defencePoints**).
  - If the armor is not worn, it does not modify any fields and returns the parameter **points**.
- The method **void upgrade()** that increases the **defencePoints** by 100.

- (10 pts) All **Weapon** objects have:
  - A field **damageAmount** of type double
    Keeps the damage points given to an enemy each time the weapon is used
  - A constructor
    Weapon(double price, double weight, double damage)
    The **damageAmount** cannot be less than zero. If it is less than zero, assign it to zero.
  - **Getter** method for the **damageAmount** field
  - A **toString()** method that returns the **price**, **weight** and **damageAmount** information as a String.

- (15 pts) All **Sword** objects have:
  - A constructor
    Sword (double price, double weight, double damage)
  - The method **boolean useItem()** that simulates the sword activity. This method prints: "The Sword is used and caused damage of X points." where X is the **damageAmount** of the Sword, and returns **true**.
  - A **toString()** method that returns a **String** stating the type of the Weapon as Sword, the Weapon's the price, weight and damageAmount.

- (25 pts) All **Bow** objects have:
  - A field **numArrows** of type **int**
  - A constructor
    Bow(double price, double weight, double damage, int numArrows)
    The **numArrows** cannot be less than zero. If it is less than zero, assign it to zero.
  - **Getter** method for the **numArrows** field
  - The method **boolean useItem()** that simulates the arrow activity.
    - If **numArrows** is greater than zero, it decrements **numArrows** by one. It also prints: "The Bow is used to throw an arrow and caused damage of X points." where X is the **damageAmount** of the Bow and returns **true**.
    - Otherwise (if **numArrows** is less than or equal to zero), it returns **false**.
  - The method **void upgrade()** that increases **numArrows** by 10.
  - A **toString()** method that returns a **String** stating the type of the Weapon as Bow, the number of arrows, the Weapon's price, weight and damageAmount.

- (10 pts) All **Upgradable** objects (i.e. Bow and Armor) are required to have the following method:
    - **void upgrade()**
      This method upgrades the item by protecting or hitting more.

You are provided a test class named as "Test.java". This class:

- Creates armor, sword and bow objects and prints the information you defined in their **toString()** methods.
- Prints the total number of items created by using the static method **getItemQuantity()**.
- Calls **useItem()** and **upgrade()** (this applies for only armor and bow objects) methods and prints the item information after these operations.

When you complete coding the Item type hierarchy, uncomment the **main** method in **Test.java** and run your program. A sample output is given below:

```
Total number of items before creating items: 0
Total number of items after creating items: 3

Armor price: 100.0
Armor weight: 50.0
Armor defense points: 200.0

Weapon type: Sword
Weapon price: 50.0
Weapon weight: 20.0
Weapon damage amount: 40.0

Weapon type: Bow
Number of arrows: 5
Weapon price: 20.0
Weapon weight: 5.0
Weapon damage amount: 2.0

Armor price: 100.0
Armor weight: 50.0
Armor defense points: 150.0

Armor is upgraded
Armor price: 100.0
Armor weight: 50.0
Armor defense points: 250.0

The Sword is used and caused damage of 40.0 points.
Weapon type: Sword
Weapon price: 50.0
Weapon weight: 20.0
Weapon damage amount: 40.0

The Bow threw an arrow and caused damage of 2.0 points
The Bow threw an arrow and caused damage of 2.0 points
The Bow threw an arrow and caused damage of 2.0 points
The Bow threw an arrow and caused damage of 2.0 points
The Bow threw an arrow and caused damage of 2.0 points
Weapon type: Bow
Number of arrows: 0
Weapon price: 20.0
Weapon weight: 5.0
Weapon damage amount: 2.0

Bow is upgraded
Weapon type: Bow
Number of arrows: 10
Weapon price: 20.0
Weapon weight: 5.0
Weapon damage amount: 2.0
```