

CS 1331 Programming Quiz 2 - Wizardry

June 27, 2017

Important Note: After you submit your quiz make sure you bring your buzzcard to the front of the room and get checked off by a TA. If you fail to do so your quiz will **NOT** be graded.

Problem description

Welcome to the Wizarding World of Harry Potter! In this world there are two kinds of people: Muggles and Wizards. Wizards are those born with magical abilities while Muggles are normal people with no magical abilities.

You have been provided with a few files, some of which are incomplete. Let's review them:

House.java

This enum represents the four houses of Hogwarts, a top-notch school of wizardry. This file has been provided for you and should **NOT** be edited.

Magical.java

This interface is implemented by all magical beings **including Wizards**. This interface has been written for you and should **NOT** be edited.

- `public abstract void castSpell(String spell):` method, should remain abstract in this interface. When this method is implemented: If the spell taken in is "expecto patronum" the method should print "name casts spell and a patronus patronus appears!". For any other spell the method should print "name casts spell!".

Person.java

This class represents a general Person. **This class should not be instantiable.**

- `private String name:` variable representing the name of the Person.
- `public Person(String name):` constructor that takes in and sets the name instance variable.
- `public String toString():` method that returns "My name is name". This method has been provided for you and should **NOT** be edited.
- `public String getName():` method that returns name. This method has been provided for you and should **NOT** be edited.

Muggle.java

This class represents a Muggle, a person with no magical ability. This is a subclass of Person.

- `private String occupation:` variable representing the occupation of the muggle.

- `public Muggle(String name, String occupation):` constructor that takes in and sets the name and occupation of the Muggle.
- `public String toString():` method that returns "My name is `name` and I am a(n) `occupation`". This method should be written to maximize code reuse.

Wizard.java

This class represents a Wizard, a person with magical abilities. This is a subclass of Person. A wizard is Magical and therefore must implement the interface Magical. **A wizard should also be Comparable to another wizard based on skill.**

- `private House house:` variable representing to which of the four houses this Wizard belongs.
- `private int skill:` variable representing the skill level of this Wizard.
- `private String patronus:` variable representing the patronus of this Wizard.
- `public Wizard(String name, House house, int skill, String patronus):` constructor that takes in and sets the instance variables.
- `public void duel(Wizard w):` method that prints "`name1` triumphed over `name2` in a duel!". The name of the wizard with the higher skill level should be `name1` and the other wizard's name should be `name2`. If the skill levels are equal then this method should print "`name3` tied with `name4` in a duel!" where `name3` is the name of this wizard and `name4` is the name of the wizard taken in as an argument.
- `public String toString():` method that returns "My name is `name` and I am a wizard in the `house` house.". This method should be written to maximize code reuse.
- properly overridden `equals` method that checks equality based on `skill` and `name`.
- Any other inherited abstract methods.

Tester.java

This class is for testing your code. Feel free to edit this file as you wish. **This file will not be graded.**

Tips

- You may need to edit the class headers.
- If there is some section of code that does not compile and you cannot fix it in time, comment it out. Partial credit is better than no credit!

Clarifications

Points will not be taken off for checkstyle and you do not need to write javadocs for your code.

You should not import any libraries or packages that trivialize the assignment. This includes data structures other than arrays (so no `ArrayList`, `List`, `Map`, `Set`, etc). `System.arraycopy` is not allowed. If you are unsure of whether something is allowed, raise your hand and ask a TA. In general, if something does a large part of the assignment for you, it is probably not allowed.

The only websites you are permitted to use are T-Square (<https://t-square.gatech.edu>) to **download and submit the quiz** and the Oracle Java API (<https://docs.oracle.com/javase/8/docs/api/>). **You may not use any other references including class notes, powerpoints, or code not provided with the quiz.**

Compiling and Testing your code

Compile your code from the `PQ2` directory with the command `javac *.java`. Test your code with the command `java Tester`. Verify your output by checking the comments in the `Tester.java` file. This tester is not exhaustive so feel free to edit it.

Turn-in Procedure

Submit your `PQ2.zip` file on T-Square as an attachment. This zip file should contain `House.java`, `Person.java`, `Muggle.java`, `Wizard.java`, and `Magical.java`. When you're ready, double-check that you have submitted and not just saved a draft.

After you submit your quiz make sure you bring your buzzcard to the front of the room and get checked off by a TA. If you fail to do so your quiz will NOT be graded.

Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your quiz files were truly submitted correctly, the upload was successful, and that your program runs with no syntax or runtime errors. It is solely your responsibility to turn in your quiz and practice this safe submission safeguard.

- After submitting the files to T-Square, return to the Assignment menu option and this quiz. It should show the submitted files.
- Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
- Re-run and test the files you downloaded from T-Square to make sure it's what you expect.
- This procedure helps guard against a few things.
 - It helps insure that you turn in the correct files.
 - It helps you realize if you omit a file or files. Missing files will not be given any credit, and non-compiling/non-running solutions will receive few to zero points. Also recall that late quizzes will not be accepted regardless of excuse. (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
 - Helps find syntax errors or runtime errors that you may have added after you last tested your code.