Lab Exam Set 2
10 points
Time: 1.5 hours

Welcome to the Triwizard Tournament, a prestigious magical competition held between the three largest wizarding schools. As a participant in this year's tournament, you will need to demonstrate your magical prowess by completing a series of challenging tasks. To help you prepare, you will create a Java program that simulates various aspects of the tournament.

You will complete this lab exam in 1.5 hours and submit your Jar file on Canvas.
Your lab instructors will download your submission and grade it.

The lab exam will be assessed on a 10-point scale using the following rubric.

10	8	6
The code is complete 100% and submitted JAR file, compiles and runs perfectly	The code is complete 90%-100%. The submitted code compiles but produces the wrong output. Or it does not compile because of minor errors.	The code is complete 60%-90%. But it does not compile and has errors.

4	2	0
The code is completed 40%-60%. But it does not compile and has errors.	The code is completed by 20%-40%. But it does not compile and has errors.	No submission or less than 20%

Setting up your Programming Environment

Create your Java project with the name "labExam1-s2"
Create a package called "tournament" in the project.

Copy starter file(s) in the package.

Let's Program

Section 1: Implementing Magical Challenges (30 minutes)

In this section, you create an interface called **MagicalChallenge**. In the interface, define the getter and setter methods, getName and setName. Additionally, define performChallenge, which has no parameters and does not return anything.

Implement the interface in the two classes below. The performChallenge method should print a short descriptive message like "Facing the mighty dragon…"

- **LakeChallenge** - also has int variable "depth", and its getter method
- **DragonChallenge** - also has double variable "difficultyLevel", and its getter method

In both classes, include a parameterized constructor to initialize both its name and its unique instance variable.

Section 2: Implementing Champion Comparisons (30 minutes)

Next, you will implement the **Champion** class with three instance variables: String name, String school, and int totalScore, alongside a parameterized constructor and setters and getters.

This class overrides **compareTo** to allow for comparisons based on champions' total score.

Additionally, override the **equals** method to check for equality based on both name and school. (compareTo method should be called in equals method)

Section 3: File Operations and Recursion (30 minutes)

In this section, you will implement the class **TournamentManager** with an ArrayList instance variable called "champions"

This class has a static method "readChampionData" that receives a file name as a String. Each line in the file contains a champion name, school, and score separated by "," (a comma). Create a Champion object for each line and add it to the ArrayList "champions".

Additionally, there is a static method “searchChampion” that receives an integer for score. First, make sure the array list is sorted using getSortedChampions. Then, use a helper method to code a recursive binary search that efficiently finds a champion by their totalScore and returns the Champion.

In the main method of this class, searchChampion(350) should return the correct name (Harry Potter).

Submission

Export your work as a jar file called “labExam1-s2.jar”. It must contain all the .java files (source files) that you developed.
When you’re satisfied you have created the correct jar file, upload it on Canvas.