CSCD 211
Lab 1

PROGRAM SPECIFICATIONS

Currently the World Cup is all the rage, so we should jump on the band wagon and show interest in the World Cup. Why not do that with a programming assignment.

I have provided a CSCD211Lab1.java class with a main method and other methods stubbed out. You CANNOT change my main method. You will need to add code to the other stubbed out methods.

For this lab we have an array of Team objects. The specifications for the classes and the files are specified below. I have provided an output run saved as individual files.

CLASS SPECIFICATIONS

Team

A Team class contains the following attributes as private instance fields:

- A code that represents the 3 letter identification – String
- The name of the country – String
- An array of Player objects

The Team class will have the following methods

- A private empty default constructor
- An explicit value constructor (passed the above instance fields
- A toString (returns a String formatted as follows:
  Country: The name of the country
  Each individual player one per line
- A compareTo ensures Team object is passed in – compares based on **the 3 letter code**
- Other methods you deem necessary.

Player

A **Player** class contains the following attributes as **private** instance fields:

- Name – String
- Position - String
- Jersey Number - int

The class should have the following instance behaviors:

- A private empty default constructor
- An explicit value constructor (passed the above instance fields)
- A toString (returns a String formatted as follows:
  The name – position space jersey number
- A compareTo method ensures Player is passed in - compares based on **the number**
- Other methods as you deem necessary*

INPUT FILE SPECIFICATIONS

Teams File

The input file is composed of multiple text files. The file that contains the codes and the team names is found within the folder files and is named teams.txt. You are guaranteed this file will have at least one entry. For this lab I have provided a constant that is set to 32. There are 32 teams in the world cup and this will be the value used for grading. You will need to prompt the user for this filename, including the folder name (testing it will not be files). The format of this file is 3 letter code,country.

Example teams.txt:

alg,Algeria
arg,Argentina

Players File

Each Team's players will be read from a file – each file is named the 3 letter code.txt (Example usa.txt). This file also resides in the same folder as teams.txt. You will need to dynamically build this file name for each team. There are a variable number of players in each file and there are blank lines between player positions in the file. You can check for a blank line by checking equals against **""**. You are guaranteed at least one entry in each file. You are guaranteed that there are no extra blank lines at the end of the file. Note: There are country specific characters in the file, don't worry about them, just read them as a normal String.

Example from arg.txt:

```
(20)  MF  Dario Vidoši          ##  22, Sion (SWI)
(21)  MF  Massimo Luongo        ##   1, Swindon Town (ENG)
(23)  MF  Mark Bresciano        ##  73, Al-Gharafa (QAT)

 (4)  FW  Tim Cahill            ##  68, New York Red Bulls (USA)
 (7)  FW  Mathew Leckie         ##   7, FSV Frankfurt (GER)
 (9)  FW  Adam Taggart          ##   4, Newcastle Jets (AUS)
```

CSCD211Lab1

Methods that you will need to write/modify.

- A `fillTeamsArray` method that is passed a Scanner Object for the keyboard.

  The method should:

  - Prompt the user for the teams filename (including the folder)
  - Opens that file after ensuring it exists – if not reprompt
  - Reads the 3 letter code, country name, calls readPlayer to build the array of players
  - Creates a Team object and places it in the array

- A `readPlayers method` that is passed a filename for the countries players.

  This method should:

  - Opens that file after ensuring it exists – if not prompt
  - Count the number of players
  - Close the file and reopen
  - Read the file – create the array of Players and return the array

- A `menu` method – reads the user's choice and verifies it is within range.

- An `executeChoice` method that is passed the choice and the Team array. This method will allow the user to execute the choice from the menu. The menu choices are specified below

- A `printArray` method that is passed a PrintStream and an array of type Team.
  - The PrintStream object will either be System.out or a reference to a file. The method should:
  - Write the Team information to the PrintStream (make sure you use the toString method provided by the Team & Player class for this)

- Helper methods as you deem necessary to aid you

MENU SPECIFICATIONS

1. Print all Teams to the screen
2. Print all Teams to the User Specified file
3. Sort the Teams by "Natural Order" (Hint: compareTo)
4. Sort the by Team Country Name (Hint: Comparator)
5. Sort each Team's Players by Number (Hint: Player compareTo)
6. Sort each Team's Players by Position (Hint: Comparator)
7. Print a entire team and only that team to a user specified file
8. Quit

OTHER SPECIFICATIONS

- You must validate all ranges for the menu
- You must ensure all files names exist
- You should use your FileUtil appropriately.
- You may use Arrays.sort for Natural Order and Comparator Sorting.
- Most of your work will occur in the executeChoice method
- For the Comparators you can write your own classes and give them whatever name you want.

TO TURN IN

A **zip** file that contains:

- All java source files needed to compile your code
- All input and output files
- Everything so we can accurately test your lab.

Name your zip file: your last name first letter of your first name lab1.zip
(Example: steinerslab1.zip)

Get started ASAP!