CIS 315 - Project 3

Project Objective: This project is designed to give the student practice in the implementation of a program that demonstrates an understanding of UML diagrams, classes, methods, inheritance and polymorphism in Java.

Problem Description: Use Java techniques to write an application program for the following.

Summit ESPORTS

Are you looking for a way to take your passion for Gaming to the next level? Look no further than Summit Esports Gaming! With a membership, you'll have access to all the exclusive events including esports leagues, tournaments, Team Play and Open Play. Also, you will get discounted rates on birthday party packages as well as other special events. As a member, you'll have access to all our top-of-the-line games. Join the ultimate gaming community at Summit Esports Gaming!

Summit Esports needs to keep track of their gamers and the tournaments that each gamer is playing via an application. This application will include the data of each player along with the gameID and the average score that they obtained during the tournaments. Tournament hosts are located at different locations.

You will have to create 3 classes that include one parent class and two child classes. The parent class is the *Person* class. The child classes are *Gamer* and *TournamentHost*.

The two methods in *TournamentHost* class will indicate the following.

1. *addGame*() → return false if the *gameID* exists.
2. *removeGame*() → return false if the *gameID* does not exist.

   Each of the classes must be able to display details such as the name and address of the *Person*, *Gamer* and *TournamentHost*.

1. The *Person* class must include:
   : the name and address of the *Person*.
   : private instance variables.
   : constructs a *Person* instance with the given name and address.
   : the getters and setters.

2. The *Gamer* class must include:
   : private instance variables showing the number of games played so far, the game codes, and the score for the corresponding game codes. The maximum number of games a player is allowed to play can be set to 20.

   : Constructs a Gamer instance with the given name and address.

Parent class properties such as the name and address can be included here.
Initialize the number of games.
Use arrays to initialize and populate the game codes keeping in mind the maximum number of games that a gamer can play. The scores must also be maintained for each game played by the gamer.

: *addGameScore*() - Adds a *gameID* and its score into the arrays.
increment the number of games after each completed game.

: *printScore*() - Prints all *gameID's* and the scores.

: *getAverageScore*() - Computes the average game score.

3. *TournamentHost* class must include:
   : the properties of the *Person* class.
   : private instance variables such as the number of games playing currently and the *gameIDs*

   : Constructs a *TournamentHost* instance with the given *name* and *address*
   initialize the number of games.
   initialize and populate the score for each game into an array.

   The maximum number of games for each tournament is 5.

   : *addGame*() - Adds a game. Returns false if the game already exists.
   The method will also track the games played.

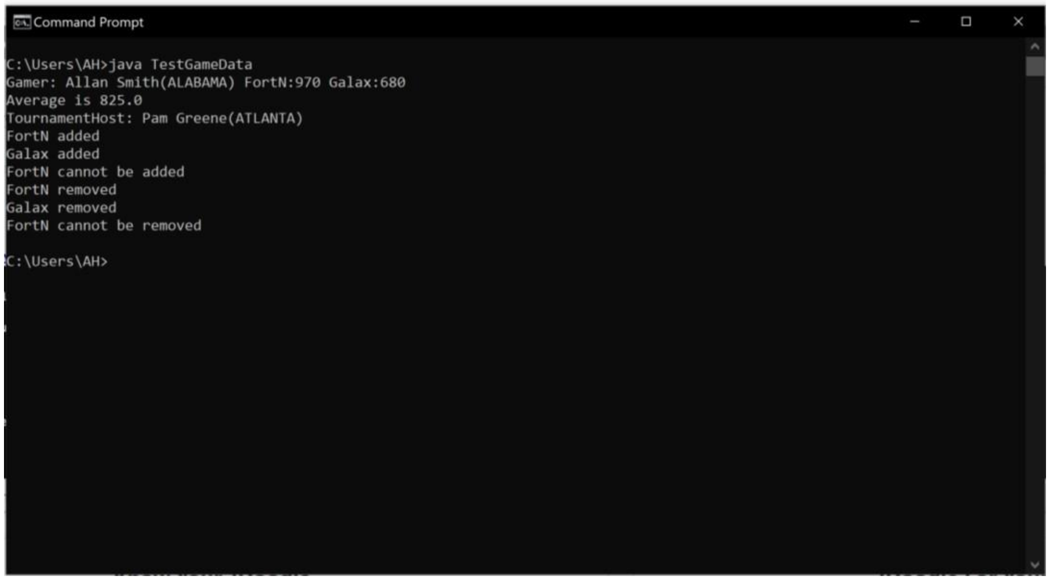
   : *removeGame*() - Removes a game. Returns false if the game cannot be found in the game list. This can be done by searching for the game index.
   If the *gameID* is found, remove the *gameID* from the list and re-arrange the *gameID* array.
   Decrease *gameID* array after a game is removed.

4. Test/Driver Class for *Person* and the subclasses:
   : class name : TestGameData
   : main method
   : test the Game class by creating a Game object
   with name and address.
   add two *gameID's* and the game scores for this, using the object created.
   **** you can hard code this or get user input.**
   print the data using *printScore*() method.
   find the average of the scores of the games played.

   : test the *TournamentHost* class by :
   Creating a *TournamentHost* object.
   populate the *gamesID* array with 3 *gameIDs*.

**use the *addGame*() and let the user know that the *gameID's* have been added.
If the same *gameID* is repeated then let the user know that
the *gameID* cannot be added.
use the *removeGame*() to remove the *gameID's*. If the same *gameID* is
given to remove, inform the user that it cannot be removed.**

Sample Output:



```
C:\Users\AH>java TestGameData
Gamer: Allan Smith(ALABAMA) FortN:970 Galax:680
Average is 825.0
TournamentHost: Pam Greene(ATLANTA)
FortN added
Galax added
FortN cannot be added
FortN removed
Galax removed
FortN cannot be removed

C:\Users\AH>
```

Submit your UML diagram and Project file(s) on Canvas using the class, method, and variable names listed in the project details.

Remember to include the project header details when submitting the project.