Middle East Technical University　　Department of Computer Engineering

CENG 443

Introduction to Object Oriented Programming Languages and Systems

Fall 2020-2021

Homework 3 - Autochess Database

Due date: 30 01 2022, Sunday, 23:59

1 Objective

This assignment aims to familiarize you with Java 8 Streams. You need to implement a database searching system consisting of four different type of entries. In this assignment you will use Java 8 Streams and lambda functions.
Keywords— Streams, Lambda Functions

2 Problem Definition

An autochess game called Dota Underworlds provided you with dataset consisting of 4 different csv files. These files contain representation of classes called **Hero**, **HeroAlliance**, **Alliance**, and **Player**. You will implement the parse function to parse the csv files to fill the lists inside **Database** class. After parsing the files you will implement five methods on the Database class that will access these lists to either return information or print something to the screen. These classes are provided to you. Details of the classes are given below:

- **Hero:** The heroes are used to fight in the autochess game. Hero class holds the statistics of the heroes based on its level. There are three levels of strength of each hero. The fields of the hero is pretty explanatory and can be seen from the provided file.
- **Alliance:** The heroes can form alliance based on their attributes. If two different heroes have the same attributes, they can form alliance and gain bonuses. To form an alliance, the player must hold required number of distinct heroes. The alliances can have different levels between one-to-three. Similarly, the required number of distinct heroes can be between one-to-three as well.
- **HeroAlliance:** The heroes have five tiers based on their strength and cost. The tiers range from one to five. The names of the attributes of the heroes are also stored on this class. They can have two or three attributes.
- **Player:** Players hold heroes in their hands and use them on a board similar to a chess board to fight opponents. This class stores the names and heroes of each player described in the csv file.

3 Input Specifications

There are four CSV files you will load the objects from. The files for heroes, alliances and hero-alliances can be considered static. However, the player filename will be given as a argument in the parse function. If the function is called again and the initial three is already parsed, there is no need to parse them again. You will only need to reload the players. The details of the CSV files given below:

- `herostats.csv` holds the information about heroes in the following format at each line:

  ```
  HeroName,Level,Health,Mana,DPS,Damage-Min,Damage-Max,Attack Speed,Move Speed,Attack Range,
  Magic Resist,Armor
  ```

- `alliances.csv` hold the information about the alliances in the following format at each line:

  ```
  AllianceName,RequiredHeroCount,LevelCount
  ```

- `heroalliances.csv` hold the information about the hero-alliances. The heroes that have two attributes is given in the following format:

  ```
  HeroName,Tier,1. AllianceName,2. AllianceName
  ```

  The heroes that have three attributes is given the following format:

  ```
  HeroName,Tier,1. AllianceName,2. AllianceName,3. AllianceName
  ```

  The alliance names is given in arbitrary order.

- The csv hold the information about the players will be given as argument. Each line is given in the following format:

  ```
  PlayerName,HeroName|Level,HeroName|Level,...,HeroName|Level
  ```

  Different from previous classes, Player class holds Hero objects. This means that you need find the hero with the corresponding name and level and add it to the list on the Player object during parsing.

Each of the information has a corresponding field on the given class.

4 Implementation Specification

You are required to implement five methods on the Database class in addition to the parser function. The details of the function are given below:

- **List<Hero> getHeroesOfParticularAlliance(String alliance, int count)**: This method should find the heroes that have the alliance attribute using the HeroAlliance list. It should sort them according to their DPS and there should be no duplicate heroes whose name is same. It should only return `count` number of heroes that is given as the second argument. For example if we print the results of the following call `Database.getInstance().getHeroesOfParticularAlliance("Mage", 3)`:

```
Hero{name='Lina', level=3, mana=75, DPS=193, damageRange=260-280,
attackSpeed=1.4, moveSpeed=290, attackRange=4, magicResist=0, armor=5}
Hero{name='Rubick', level=3, mana=100, DPS=169, damageRange=200-240,
attackSpeed=1.3, moveSpeed=310, attackRange=2, magicResist=20, armor=5}
Hero{name='Puck', level=3, mana=60, DPS=155, damageRange=240-288,
attackSpeed=1.7, moveSpeed=290, attackRange=3, magicResist=0, armor=5}
```

- **public Map<Integer, List<HeroAlliance> > getHeroAllianceMatchingTier(int allianceRequiredCount, int allianceLevelCount)**: This method should find the Alliances that have and that have the given required count and level count. After finding them, it find the HeroAlliance objects that have any of those Alliances and group them according to their tier. For example if we print the results of the following call `Database.getInstance().getHeroAllianceMatchingTier(2, 2)`

```
Tier: 1
    HeroAlliances:
    HeroAlliance{name='Anti-Mage', tier=1, alliances=[Hunter, Rogue]}
    HeroAlliance{name='Batrider', tier=1, alliances=[Troll, Knight]}
    HeroAlliance{name='Bounty Hunter', tier=1, alliances=[Assassin, Rogue]}
    HeroAlliance{name='Dazzle', tier=1, alliances=[Poisoner, Troll, Healer]}
    HeroAlliance{name='Drow Ranger', tier=1, alliances=[Vigilant, Hunter, Heartless]}
    HeroAlliance{name='Enchantress', tier=1, alliances=[Shaman, Healer]}
    HeroAlliance{name='Slardar', tier=1, alliances=[Warrior, Scaled]}
    HeroAlliance{name='Snapfire', tier=1, alliances=[Brawny, Dragon]}
    HeroAlliance{name='Venomancer', tier=1, alliances=[Summoner, Poisoner, Scaled]}
Tier: 2
    HeroAlliances:
    HeroAlliance{name='Bristleback', tier=2, alliances=[Brawny, Savage]}
    HeroAlliance{name='Juggernaut', tier=2, alliances=[Brawny, Swordsman]}
    HeroAlliance{name='Luna', tier=2, alliances=[Vigilant, Knight]}
    HeroAlliance{name='Meepo', tier=2, alliances=[Summoner, Rogue]}
    HeroAlliance{name='Nature's Prophet', tier=2, alliances=[Summoner, Shaman]}
    HeroAlliance{name='Spirit Breaker', tier=2, alliances=[Brute, Savage]}
    HeroAlliance{name='Windranger', tier=2, alliances=[Vigilant, Hunter]}
Tier: 3
    HeroAlliances:
    HeroAlliance{name='Alchemist', tier=3, alliances=[Brute, Poisoner, Rogue]}
    HeroAlliance{name='Beastmaster', tier=3, alliances=[Hunter, Brawny, Shaman]}
    HeroAlliance{name='Lifestealer', tier=3, alliances=[Brute, Heartless]}
    HeroAlliance{name='Lycan', tier=3, alliances=[Human, Summoner, Savage]}
    HeroAlliance{name='Omniknight', tier=3, alliances=[Human, Knight, Healer]}
    HeroAlliance{name='Shadow Shaman', tier=3, alliances=[Summoner, Troll]}
    HeroAlliance{name='Slark', tier=3, alliances=[Assassin, Scaled]}
    HeroAlliance{name='Treant Protector', tier=3, alliances=[Shaman, Healer]}
Tier: 4
    HeroAlliances:
    HeroAlliance{name='Doom', tier=4, alliances=[Brute, Demon]}
    HeroAlliance{name='Lone Druid', tier=4, alliances=[Summoner, Savage, Shaman]}
    HeroAlliance{name='Mirana', tier=4, alliances=[Vigilant, Hunter]}
    HeroAlliance{name='Sven', tier=4, alliances=[Knight, Swordsman, Rogue]}
    HeroAlliance{name='Templar Assassin', tier=4, alliances=[Vigilant, Assassin, Void]}
    HeroAlliance{name='Tidehunter', tier=4, alliances=[Warrior, Scaled]}
Tier: 5
    HeroAlliances:
    HeroAlliance{name='Axe', tier=5, alliances=[Brute, Brawny]}
    HeroAlliance{name='Medusa', tier=5, alliances=[Hunter, Scaled]}
    HeroAlliance{name='Troll Warlord', tier=5, alliances=[Warrior, Troll]}
```

- **public List<List<Hero> > getPlayerHeros(int mana, int health, int moveSpeed)**: This method should return the heroes of each player if the hero has mana, health and move speed larger than the arguments. It should values larger than all of them. If the player has no such hero, it should return empty for that player. For example if we print the results of the following call `Database.getInstance().getPlayerHeros(80, 1800, 300)`

```
============================Player1============================
==============================================================
============================Player2============================
==============================================================
============================Player3============================
==============================================================
============================Player4============================
Hero{name='Legion Commander', level=3, mana=100, DPS=176, damageRange=200-240,
attackSpeed=1.25, moveSpeed=315, attackRange=1, magicResist=0, armor=5}
Hero{name='Legion Commander', level=3, mana=100, DPS=176, damageRange=200-240,
attackSpeed=1.25, moveSpeed=315, attackRange=1, magicResist=0, armor=5}
Hero{name='Legion Commander', level=3, mana=100, DPS=176, damageRange=200-240,
attackSpeed=1.25, moveSpeed=315, attackRange=1, magicResist=0, armor=5}
Hero{name='Lycan', level=3, mana=100, DPS=240, damageRange=240-288,
attackSpeed=1.1, moveSpeed=315, attackRange=1, magicResist=0, armor=5}
Hero{name='Lycan', level=3, mana=100, DPS=240, damageRange=240-288,
attackSpeed=1.1, moveSpeed=315, attackRange=1, magicResist=0, armor=5}
Hero{name='Lycan', level=3, mana=100, DPS=240, damageRange=240-288,
attackSpeed=1.1, moveSpeed=315, attackRange=1, magicResist=0, armor=5}
Hero{name='Spirit Breaker', level=2, mana=100, DPS=187, damageRange=110-140,
attackSpeed=0.67, moveSpeed=330, attackRange=1, magicResist=0, armor=5}
Hero{name='Spirit Breaker', level=2, mana=100, DPS=187, damageRange=110-140,
attackSpeed=0.67, moveSpeed=330, attackRange=1, magicResist=0, armor=5}
Hero{name='Spirit Breaker', level=2, mana=100, DPS=187, damageRange=110-140,
attackSpeed=0.67, moveSpeed=330, attackRange=1, magicResist=0, armor=5}
==============================================================
=============================Player5=============================
Hero{name='Void Spirit', level=3, mana=100, DPS=300, damageRange=280-320,
attackSpeed=1.0, moveSpeed=305, attackRange=1, magicResist=0, armor=5}
================================================================
...
...
...
=============================Player47=============================
================================================================
=============================Player48=============================
================================================================
```

- **public void printAverageMaxDamage(int minDamage)**: This method should calculate and print the average of the maximum damage field of the heroes of each player if the hero has minimum damage larger than the argument. The output of a example call `Database.getInstance().printAverageMaxDamage(60)`:

  ```
  119.0
  125.45454545454545
  204.0
  187.2
  205.0
  159.375
  224.44444444444446
  202.5
  116.35714285714286
  188.8
  286.6666666666667
  338.4
  233.57142857142858
  154.85714285714286
  ...
  ...
  ...
  246.15384615384616
  188.30769230769232
  187.52941176470588
  208.57142857142858
  ```

- **public void printAlliances(String[] alliances, double attackSpeed)**: This method should print the heroes of players if the player and the heroes satisfy certain conditions. The condition is, the heroes of a player belongs to one of the alliances given as the first argument and its attack speed smaller than or equal to the second argument. If none of the heroes of a player satisfy this condition, that player will not be printed. Moreover, only the heroes that satisfy these two conditions should be printed. The output of a single player of this function should be in this format:

  ```
  <PlayerName>
  Name: <Hero1Name> Level: <Hero1Level>
  Name: <Hero2Name> Level: <Hero2Level>
  ...
  Name: <HeroNName> Level: <HeroNLevel>
  ```

  Example output of `Database.getInstance().printAlliances(new String[]"Mage", 1.3)` can be seen below:

  ```
  Player16
  Name: Rubick Level: 3
  Player36
  Name: Rubick Level: 2
  Name: Rubick Level: 2
  Player37
  Name: Rubick Level: 1
  ```

All of the examples involving Players acquired from parsing the `players.csv` file provided to you.

4.1 Grades:

Grades schema of the homework is given below:

- **getPlayerHeros**: 10 pts
- **printAverageMaxDamage**: 10 pts
- **getHeroAllianceMatchingTier**: 15 pts
- **getHeroesOfParticularAlliance**: 15 pts
- **printAlliances**: 30 pts
- **Rest**: 20 pts

5 Specifications

- Your code must be written in Java (version 8).
- Late Submission is allowed. A penalty of $5 \times Lateday^2$ will be applied for submissions that are late at most 3 days.
- The functions `printAlliances` and `printAverageMaxDamage` prints their results to the standard output(stdout).
- You will only complete parts where TODO comments are.
- You are only allowed to declare local variables in the methods, i.e., do not declare new attributes/methods in class Database. You can also override equals method on the classes if you feel the need.
- Do not use loops or recursion instead of Java 8 Streams. If you do, you will not gain any points from that particular function.
- Submissions will also be evaluated with white box to make sure that your code adheres to Object Oriented Programming principles.
- Everything you submit should be your own work. Usage of binary source files and codes found on internet is strictly forbidden.
- Please follow the course page on ODTUClass for updates and clarifications.
- Please ask your questions related to homework through ODTUClass instead of emailing directly to teaching assistants if it does not involve code or solution.

6 Submission

Submission will be done via ODTUClass. Create a zip file named `eXXXXXXX.zip` (eXXXXXXX is your student id) that contains all of the modified java source code files without any directory. If you did not modify a class provided to you, there is no need to put them in the zip file. They should be able to compile with:

```
> javac *.java
```