ENSF 380

Exercises - Lesson 14

The following exercises correspond to the video for Lesson 14.

Exercise 14.1

1. Implement a Character ArrayDeque.
2. Convert a Character array to ArrayDeque.
3. Use several different methods to add elements to the collection.
4. Use several different methods to inspect and print elements, including descendingIterator.
5. Use several different methods to remove elements from the collection.
6. Check your work against the example in the repository.

```
Lesson14_DataStructures/03_ArrayDeque
```

- What is ArrayDeque good at?
- What is ArrayDeque bad at?
- When would you use ArrayDeque?

Exercise 14.2

1. Read the documentation for HashMap.
2. Look at the code in the repository.
3. Without running the code, determine what the output should be.

```
Lesson14_DataStructures/04_HashMap
```

- Were you able to predict the output based on the documentation?
- If you were to add HashMap to the reference table presented earlier, what values would you fill in for order maintained, duplicates, allows null, and thread-safe?
- When would you use a HashMap?

Exercise 14.3

1. Implement the program shown in the UML diagram in the repository.
2. Use the code in `MyOutput.java` to test your code.
3. Compare your output with the output in `output.txt`.
4. Each class should be public and placed in its own file.
5. You may modify the MyOutput class; this is not tested.

Lesson14_DataStructures/06_RobotData

- Assumptions
  - Each valid robot data line consists of a valid robot ID, a valid date, a valid movement made up of an action and direction abbreviation, and a valid sensor in brackets, as shown in the example main.
  - A valid robot ID always consists of three digits followed by a capital letter.
  - A valid sensor can consist of any single English word (lowercase letters only, no hyphens, spaces or numbers).
  - The only valid actions and directions are contained in the provided enumerations.
  - You should test for the validity of each element. If the robot ID, date, action, direction or sensor are invalid, you should throw an IllegalArgumentException. These exceptions should be passed up to container classes, and therefore caught in RobotDataRecord. As a catch behaviour, RobotDataRecord should skip any data lines that contain invalid information.
  - When cloning, you do not need to create a deep copy of the LocalDate object. All other elements should be cloned.
- If your code is working properly, it will produce exactly the same output as shown in the output file, provided you do not change the example main.

Tip: Look at the regular expressions for each class in order to determine what the String which is sent to the constructor should contain. The regular expressions are meant to parse the parameter(s) provided to the constructor.

Exercise 14.4

1. Extend the program shown in the UML diagram in the repository as described in the next steps. Some additional requirements are:
   - Your implementation should be consistent with the diagram, although not all parts of the diagram need to be implemented, as indicated below.
   - `CharacterHealer` is included in the UML diagram for the sake of completeness. It is not part of this exercise.
   - Do not modify or remove any existing lines of code for classes shown on the diagram, although you will need to add code. (`PlayTheGame` is a `main()` and is not part of the UML diagram. It may be modified.)
   - All quotation marks included around Strings in these instructions are intended to denote the String and are not part of the String itself.

    - Note that if you generate output from `PlayTheGame`, it may not match the example output completely, as the combat contains some randomness.

2. Implement the class `CharacterRogue`.
    - `getWeapon()` is a standard getter method.
    - The attack damage for a rogue is 10.
    - The value returned by the method `getAttackMessage()` is demonstrated in the output.
    - `talk()` returns the input String prefaced with "....(" and succeeded by ")....", without additional whitespace. For example, given the input "Hello", the method would return "....(Hello)...."
3. Implement a concrete method `asString()` for each enumeration element in the enumeration `CharacterClasses`. The method should return the enumeration name with the first letter capital, and the remaining letters lowercase. For example, calling `getCharacterClass()` when the variable `CHARACTER_CLASS` is `WARRIOR` should return "Warrior".
4. Implement the method `validateAndRecordAttackPriority()` in the class `GameCharacter`. This method must:
    - Throw the exception specified in the UML diagram when the provided argument is already stored in `attackPriorities`.
    - Store the provided argument in `attackPriorities` if not already present.
5. You may **optionally** implement `PrintOutput.java`, `PrintPlaintext.java`, and `PrintHTML.java`.
    - In the class `PrintPlaintext`,
        - The methods `printFightLog()` and `printStats()` are identical to the methods of the same name in the class `PlayTheGame`.
        - `printMessage()` should be implemented with:
          `System.out.println(message);`
    - In the class `PrintHTML`,
        - The method `printMessage()` prints the message prefaced with "<p>" and succeeded with "</p>" without additional whitespace.
        - Example output from the method `printFightLog()` is shown below.
        - Example output from the method `printStats()` is shown below.

Lesson14_DataStructures/07_Game

Example output from `PrintHTML` `printStats()`

```
<p>Name: Xena<br />
Life: 100<br />
Class: Warrior<br />
Says: I AM XENA THE WARRIOR!!!!!<br />
Attack speed: 92<br />
Damage: 12<br />
Attack: Xena attacks with their sword and short sword.</p>
```

Example output from `PrintHTML` `printFightLog()`:

```
<ul>
<li>Katniss says: GAME ON!!!!!</li>
<li>Xena says: LET'S GO!!!!!</li>
<li>Katniss attacks with their bow and knife.</li>
<li>Katniss does 9 damage.</li>
<li>Xena has 2 life remaining.</li>
<li>Xena attacks with their sword and short sword.</li>
<li>Xena does 4 damage.</li>
<li>Katniss has 17 life remaining.</li>
<li>Katniss attacks with their bow and knife.</li>
<li>Katniss does 7 damage.</li>
<li>Xena has died!</li>
</ul>
```

Tip: If you choose to complete the optional implementation, you will need to make your own `main()` or modify the existing one to observe the functionality.