JP2 2019: Lab exam (Version B)

This lab exam is intended to be completed during the 1-3pm Monday timeslot (Lab sections 3 and 4). If you are taking the exam at a different time, you should locate the correct specification instead of this one.

Overview

Your task is to develop classes to model a hide-and-seek game involving a set of players: one or more of the players is defined as a seeker, and the rest of the players try to hide from the seekers. The game also involves a set of **Hiding Locations**. In your program, every player will take turns until either a maximum number of turns has taken place, or else the seeker has found all of the other players.

A turn is arranged as follows: every player, one at a time, chooses to move to a new location or to stay in their current location. In addition, when it is a seeker's turn, if there are any other players in the same location as the seeker, those players will be "caught" – i.e., all of those players will be removed from the game.

Here is an example game, with seeker **S**, players **P1** and **P2**, and locations **Attic**, **Basement**, and **Bedroom:**

- In the initial state, **S** is in **Attic**, **P1** is in **Basement**, and **P2** is in **Bedroom**
- Turn 1:
    - **S** moves to **Basement**
    - **P1** stays in **Basement**
    - **P2** moves to **Attic**
- Turn 2:
    - **S** catches **P1** (because they are both in the same place); **S** then chooses to stay in **Basement**
    - **P1** does not take any action because they have been caught
    - **P2** stays in **Attic**
- Turn 3:
    - **S** moves to **Attic**
    - **P2** moves to **Basement**
- Turn 4:
    - **S** stays in **Attic**
    - **P2** moves to **Attic**
- Turn 5:
    - **S** catches **P2**

In this example, the game is over after 5 turns because the seeker has caught all of the other players. Depending on the choices made by each player, the game could also go on for longer and have different outcomes.

Task 1: Location (2 marks)

Note about implementation: all classes created in this exam should be put in the ***hide_seek*** *package.*

You must create an enumerated type **Location** with the following values:

BEDROOM, ATTIC, LIVING_ROOM, DINING_ROOM, BATHROOM, BASEMENT

Task 2: Player (7 marks)

You must create a class **Player** representing a player of the game including the following properties:

- name: (a String)
- active: (a boolean flag indicating whether this player is "active", i.e., has not been caught)
- seeker: (a boolean flag indicating whether this player is a seeker)
- location: (a Location instance indicating where this player is currently located)

The **Player** class should have a constructor with the following signature:

public Player(String name, boolean seeker)

In addition to setting the values of those two fields, the constructor should also set the **active** flag to **true**, and the **location** field should be set to a randomly chosen location

The **Player** class should also include:

- A complete set of **get** methods, as well as **set** methods for the **active** and **location** fields
- Appropriate implementations of **equals()**, **hashCode()**, and **toString()**

Task 3: HideAndSeekGame (4 marks)

Create a class **HideAndSeekGame** representing a game of hide-and-seek. This class should have one field:

- players: (a field representing the players in the game. You can choose whatever data type you want for this field)

You should also include a constructor that sets that field. Your constructor should throw an **IllegalArgumentException** if the provided list of players does not include at least one seeker.

You should also include a **get** method for the **players** field. **You do not need to override equals(), hashCode(), or toString() for this class**.

Task 4: playGame (6 marks)

In your **HideAndSeekGame** class, implement one additional public method, as follows:

public void playGame(int numRounds)

This method should behave as follows:

- It should loop for the indicated number of rounds
- Within each round, it should loop through all game players and do the following:
  - If the player is not active, do nothing
  - Otherwise, if the player is a seeker:
    - Find all other active players in the same location as the seeker
    - If that player is not also a seeker, deactivate them (i.e., **setActive(false)**)
  - Every player should also move to a new location at the end of each turn. (Note that the newly chosen location may be the same as the previous location – you do not need to prevent this)
- At the end of each round, your code should check whether any non-seeker players are still active; if not, the game should exit even if the number of rounds has not finished

The sample game on the previous page shows one possible way that this method could proceed.

What to submit

On Moodle, go to **Lab Exam Submission – Version B** and upload the following three files:

- Location.java
- Player.java
- HideAndSeekGame.java

Be sure to submit to the correct submission link, and be sure to submit before the deadline.