CS 249: Assignment 07

Programming Assignments (95%)

This assignment will focus on implementing the **Strategy Design Pattern** to handle movement behaviors for creatures.

This assignment will rely on code from the previous assignment(s). **Do NOT copy code from the previous assignments into this one!** Instead, make the necessary imports.

Also note: **if your code from Assignment 06 wasn't working before, it must work now!**

Player.java

Create a java file with a public class Player. **Player INHERITS from Creature**. Because Creature did not define code for draw() (and Player is a concrete class), Player will have to do so now. **Player does NOT store any instance data of its own! Instead, it relies on the data in Creature**. Also, this class does NOT have to provided code for load() (Creature has already taken care of that). This class will have the following public methods.

- **public Player()**
  - Basically does nothing.
- **public Player(int row, int col)**
  - Calls the superclass's constructor to set row and col.
- **public String toString()**
  - Returns "Player at " + getRow() + "," + getCol().
  - Example: if the Player is located at row = 6 and col = 7, then return "Player at 6,7"
- **public void draw(GameBoard map)**
  - Call setPos on map to draw an 'P' at the row and col position of the Player.

MoveAlgorithm.java

Create a java file with a public ***interface*** MoveAlgorithm. It contains ONE public ***abstract*** method:

- ***public abstract void move(Creature current, Player p);***

SeekPlayer.java

Create a java file with a public class SeekPlayer. **SeekPlayer IMPLEMENTS the interface MoveAlgorithm**. This class will have the following public methods:

- **public void move(Creature current, Player p)**
  - The goal is to move the current Creature towards the Player position.
  - Please note that the creature will NOT move diagonally (only up/down OR left/right).

    - If the creature is right on top of the player, the creature should not move at all.
    - Get the row distance and column distance between the current Creature and the Player.
    - If EITHER of these distances are not zero (positive or negative is fine):
        - If the absolute value of the distance in rows is GREATER than the absolute value of the distance in columns:
            - **If the distance is negative, move the current creature up (negative)**
            - **Otherwise, move the current creature down (positive)**
        - Otherwise:
            - **If the distance is negative, move the current creature left (negative)**
            - **Otherwise, move the current creature right (positive)**

AvoidPlayer.java

Create a java file with a public class AvoidPlayer. **AvoidPlayer IMPLEMENTS the interface MoveAlgorithm**. This class will have the following public methods:

- **public void move(Creature current, Player p)**
    - The goal is to move the current Creature AWAY from the Player position.
    - Please note that the creature will NOT move diagonally (only up/down OR left/right).
    - If the creature is right on top of the player, the creature should not move at all (cowering in fear).
    - Get the row distance and column distance between the current Creature and the Player.
    - If EITHER of these distances are not zero (positive or negative is fine):
        - If the absolute value of the distance in rows is GREATER than the absolute value of the distance in columns:
            - ***If the distance is negative, move the current creature down (positive)***
            - ***Otherwise, move the current creature up (negative)***
        - Otherwise:
            - ***If the distance is negative, move the current creature right (positive)***
            - ***Otherwise, move the current creature left (negative)***

Mover.java

Create a java file with a public ***interface*** Mover. It contains ONE public ***abstract*** method:

- ***public abstract void performMove(Player p);***

Bat.java

Create a java file with a public class Bat. **Bat INHERITS from Creature**. **Bat also IMPLEMENTS Mover.** Bat will have to store an instance of MoveAlgorithm (specifically, **AvoidPlayer**), but Bat should not have any OTHER instance data. This class will have the following public methods.

- **public Bat()**
  - If you have not created and stored an AvoidPlayer instance, do so now.
- **public Bat(int row, int col)**
  - Calls the superclass's constructor to set row and col.
  - If you have not created and stored an AvoidPlayer instance, do so now.
- **public String toString()**
  - Returns "Bat at " + getRow() + "," + getCol().
  - Example: if the Bat is located at row = 6 and col = 7, then return "Bat at 6,7"
- **public void draw(GameBoard map)**
  - Call setPos on map to draw an 'B' at the row and col position of the Bat.
- **public void performMove(Player p)**
  - Call move(this, p) on your AvoidPlayer instance variable.

Orc.java

Create a java file with a public class Orc. **Orc INHERITS from Creature**. **Orc also IMPLEMENTS Mover.** Orc will have to store an instance of MoveAlgorithm (specifically, **SeekPlayer**), but Orc should not have any OTHER instance data. This class will have the following public methods.

- **public Orc()**
  - If you have not created and stored an SeekPlayer instance, do so now.
- **public Orc(int row, int col)**
  - Calls the superclass's constructor to set row and col.
  - If you have not created and stored an SeekPlayer instance, do so now.
- **public String toString()**
  - Returns "Orc at " + getRow() + "," + getCol().
  - Example: if the Orc is located at row = 6 and col = 7, then return "Orc at 6,7"
- **public void draw(GameBoard map)**
  - Call setPos on map to draw an 'O' (letter) at the row and col position of the Orc.
- **public void performMove(Player p)**
  - Call move(this, p) on your SeekPlayer instance variable.

Thunderdome.java

This program has been provided for you. It creates a GameBoard, a Player, four Orcs, and four Bats. It then asks for character input:

- w = Move up
- s = Move down
- a = Move left
- d = Move right
- x = Exit

The Orcs will then seek the player, while the Bats will fly away.

Initial Output of Thunderdome:

```
MAP:
O............................
.............B.............O.
.............................
.O..........B.P.B............
............B................
....................O........
.............................

Enter action:
```

Testing Screenshot (5%)

Submit a screenshot showing the results of running the test program(s).

Grading

Your OVERALL assignment grade is weighted as follows:

- 5% - Testing results screenshot
- 95% - Programming assignments