**Lab 03 Piece Inheritance & Polymorphism**

**Summary:**

Using the provided zip file you will use inheritance and polymorphism to complete the code. Note, I’m not looking for you to fully write the logic and jump movement of checkers. Rather to implement logic that lets me move a piece “forward”. Keep in mind, this is relative to the pieces color. Take the following terminal output:

======================================================

\_\_R0-0\_\_\_0-1\_\_\_KR0-2\_\_\_0-3\_\_\_\_0-4\_\_\_\_0-5\_\_\_\_0-6\_\_\_\_0-7\_\_

\_\_1-0\_\_\_\_1-1\_\_\_\_1-2\_\_\_\_R0-2\_\_\_1-4\_\_\_\_1-5\_\_\_\_1-6\_\_\_\_1-7\_\_

\_\_2-0\_\_\_\_2-1\_\_\_\_2-2\_\_\_\_2-3\_\_\_\_2-4\_\_\_\_2-5\_\_\_\_2-6\_\_\_\_2-7\_\_

\_\_3-0\_\_\_\_3-1\_\_\_\_3-2\_\_\_\_3-3\_\_\_\_3-4\_\_\_\_3-5\_\_\_\_3-6\_\_\_\_3-7\_\_

\_\_4-0\_\_\_\_4-1\_\_\_\_4-2\_\_\_\_4-3\_\_\_\_4-4\_\_\_\_4-5\_\_\_\_4-6\_\_\_\_4-7\_\_

\_\_5-0\_\_\_\_5-1\_\_\_\_W0-0\_\_\_5-3\_\_\_\_5-4\_\_\_\_5-5\_\_\_\_5-6\_\_\_\_5-7\_\_

\_\_6-0\_\_\_\_6-1\_\_\_\_6-2\_\_\_\_6-3\_\_\_\_6-4\_\_\_\_6-5\_\_\_\_6-6\_\_\_\_6-7\_\_

\_\_7-0\_\_\_WRK-2\_\_\_7-2\_\_\_\_7-3\_\_\_\_7-4\_\_\_\_7-5\_\_\_\_7-6\_\_\_\_7-7\_\_

Any piece marker R is red, it can only move as the ROW number (the first number) increases. Any piece marker W is white, and can only move as the ROW number get’s smaller. A valid move for R0-0 is to row 1, or greater. While this isn’t a logical checkers move, it gets the point across of what we’re trying to achieve. These are regular pieces.

A king piece moves in any direction regardless.

I’m not looking for jump logic, and I’ve included code for you to play around with and use to test.

The code here will also error out because you need to implement the inheritance and polymorphism fully. I’ve only written the skeleton.

To ensure this was written correctly I’ve created a Piece list, and a piece\_holder. They can hold any piece, and calling .Move should run the correct Move function based on which piece you’re moving. Kings can move forward or backward, regular pieces only “forward”.

**Piece.java**

package com.company;  
  
public class Piece {  
 String piece\_name;  
 String piece\_color;  
 int x\_cord;  
 int y\_cord;  
  
 public String getPiece\_name() {  
 return piece\_name;  
 }  
  
 public void setPiece\_name(String piece\_name) {  
 this.piece\_name = piece\_name;  
 }  
  
 public int getX\_cord() {  
 return x\_cord;  
 }  
  
 public void setX\_cord(int x\_cord) {  
 this.x\_cord = x\_cord;  
 }  
  
 public int getY\_cord() {  
 return y\_cord;  
 }  
  
 public void setY\_cord(int y\_cord) {  
 this.y\_cord = y\_cord;  
 }  
  
 public Piece(String piece\_name, String piece\_color) {  
 this.piece\_name = piece\_name;  
 this.piece\_color=piece\_color;  
 }  
  
 public Piece(){  
  
 }  
}

**RegularPiece.java**

package com.company;  
  
public class RegularPiece extends Piece{  
  
  
 public void Move(Board b, int x, int y){  
 //code here  
 }  
  
 public RegularPiece(String name, String color){  
 super(name,color);  
 }  
  
}

**KingPiece.java**

package com.company;  
  
public class KingPiece extends Piece{  
  
 public void Move(Board b, int x, int y){  
 //code here  
 }  
   
 public KingPiece(String name, String color){  
 super(name,color);  
 }  
}

In previous labs the direction of movement was irrelevant; meaning forward, backward, left right, etc, didn’t matter. This time however, they will.

When you write the code to move a piece it should take into account the pieces ability to move “forward”. Remember, forward is a relative term, forward is the other players backward if you think about it.

Not only will you need to account for Piece color but if it’s a regular or king piece, A king can move forward and backward, while a regular piece can only move forward.

You DO NOT need to implement jumps, I will also not be testing “is there a piece on the space I’m moving to”. I’ll attempt to move a regular piece correctly and it should succeed. I will then attempt to move it incorrectly and it should print out a simple “failed” message.

I will do the same with the king piece, but forward and backward.

I will do this for both red and white pieces. A semi functioning example will be attached to the assignment as a zip file.