Announcements

Assignment 2

In this assignment you will write a program to play the game Connect Four.

DEADLINE
This assignment is due on Friday, April 4 at 11:00 pm.

DESCRIPTION
Connect Four is a popular game played by dropping checkers into a vertical grid with six rows and seven columns. The checkers can be dropped in any of the seven columns (permitted they are not full). Once dropped, the checker falls to the lowest unoccupied space. The winner is the first player to occupy four consecutive spaces in a straight line, either horizontal, vertical, or diagonal.

SPECIFICATIONS
In this project you will write a program to represent the game Connect Four using a two-dimensional array of integers to represent the grid and a Picture object to display the game. You will create a class called `ConnectFour` to represent the game. The class `PlayConnectFour` (provided) will allow the user to play the game. When run, this program will display the game board and ask the user to specify a column. The user is always player 1 (red checkers) and goes first. After the user enters a column, the opponent will select a column for their piece.

For this project you will create a simple naive opponent that selects a column at random. Then the program will update and repaint the board and ask the user to select another column. This will repeat until the game ends in a win, loss, or draw.

To help with this project a `Referee` class is provided. The referee will look at your grid and tell you whether the game is a win, loss, draw, or is not finished.
Dynamic Binding

- A polymorphic method
  - A method that has multiple meanings
  - Created when a subclass overrides a method of the superclass
- Late binding or dynamic binding
  - The appropriate version of a polymorphic method is decided at execution time

Dynamic Binding and Abstract Classes

- Controlling whether a subclass can override a superclass method
  - Field modifier `final`
    - Prevents a method from being overridden by a subclass
  - Field modifier `abstract`
    - Requires the subclass to override the method
- Early binding or static binding
  - The appropriate version of a method is decided at compilation time
  - Used by methods that are `final` or `static`
Dynamic Binding and Abstract Classes

• Overloading methods
  – To overload a method is to define another method with the same name but with a different set of parameters
  – The arguments in each version of an overloaded method determine which version of the method will be used

Abstract Classes

• Abstract classes
  – An abstract class is used only as the basis for subclasses
    • It defines a minimum set of methods and data fields for its subclasses
  – An abstract class has no instances
  – An abstract class should, in general, omit implementations except for the methods that
    • Provide access to private data fields
    • Express functionality common to all of the subclasses

Abstract Classes (Continued)

– A class that contains at least one abstract method must be declared as an abstract class
– A subclass of an abstract class must be declared abstract if it does not provide implementations for all abstract methods in the superclass
**Package**

- Packages allow grouping of related classes.
- Package names should be lowercase.
- For package named `mypack`, place all classes in the directory `mypack`.
- Include the package statement at the beginning of each file.
  ```java
directory
```
- Package classes can be used by importing the package.
  ```java
import mypack.*;
```

**Polymorphism**

- Trace the execution of the following code:
  ```java
import shape.*;
Circle c1 = new Circle(1.0);
System.out.println(c1);
```
- Which versions of `perimeter()` and `area()` are called?

**Object Type Compatibility**

- Example:
  ```java
import shape.*;
Shape[] shapes = new Shape[5];
shapes[0] = new Rectangle(1.0, 2.0);
shapes[1] = new Triangle(1.0, 2.0, 3.0);
shapes[2] = new Circle(1.0);
shapes[3] = new Square(1.0);
shapes[4] = new Ellipse(1.0, 2.0);
for(int i = 0; i < shapes.length; i++)
    System.out.println(shapes[i]);
A shape with perimeter = 6.0 and area = 2.0.
A shape with perimeter = 6.0 and area = 18.97.
A shape with perimeter = 4.0 and area = 1.0.
A shape with perimeter = 4.96 and area = 1.57.
```