Show all of your work clearly in the space provided or on the additional page at the end of the exam. Be sure to **read each problem carefully**. Note that the exam is double sided. You are not required to provide comments in your code or include `import` statements.

1. (10 points) Consider the following code snippet:

```java
int i = 3;
System.out.println(i++);
System.out.println(++i);
```

Identify the line of code with a pre-increment operator, and identify the line of code with a post-increment operator. Indicate what will be displayed to the console.

2. (10 points) Override the toString() in a child class so that it adds quotes to whatever the parent toString() method returns. For example, if the parent class produces Exam, your implementation should produce ”Exam”, and if the parent class produces ”I”, your implementation should produce ””I””.

3. (6 points) For each of the following expressions, indicate whether or not the predicate method (a method that returns a boolean value) will be called. If not, explain why not.

```java
true || method()
true && method()
false || method()
```

4. (20 points) Indicate whether each statement is true of false.

(a) ______ The `instanceof` operator returns a boolean value.

(b) ______ The toString() method of the Object class returns a string containing a package name.

(c) ______ A `private` instance variable declared in a superclass is not part of a subclass object.

(d) ______ A `protected` class variable is visible within the constructor of any subclass.

(e) ______ The `implements` keyword is used to indicate to the compiler that a class implements one or more interfaces.

(f) ______ The `inherits` keyword is used to indicate to the compiler that a class inherits from another class.

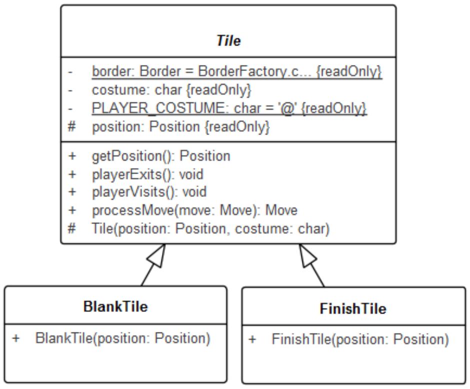
(g) ______ The @Override annotation is used to indicate to the compiler that the method implementation replaces a different implementation.

(h) ______ A class with an `abstract` method must be declared as `abstract`.

(i) ______ Every object, regardless of class, has an equals() method.

(j) ______ Interfaces cannot contain instance variables.

5. (14 points)


Tile
- border: Border = BorderFactory.c... {readOnly}
- costume: char {readOnly}
- PLAYER_COSTUME: char = '@' {readOnly}
# position: Position {readOnly}
+ getPosition(): Position
+ playerExits(): void
+ playerVisits(): void
+ processMove(move: Move): Move
# Tile(position: Position, costume: char)
BlankTile
+ BlankTile(position: Position)
FinishTile
+ FinishTile(position: Position)

Suppose that bTile is a BlankTile reference that points to a BlankTile object, and fTile is a FinishTile reference that points to a FinishTile object. Based on the above UML class diagram indicate whether each statement is true of false.

(a) ______ Tile is an `abstract` class.

(b) ______ BlankTile is a child of Tile.

(c) ______ Tile finishTile = new FinishTile(); is a legal statement.

(d) ______ Tile tile = fTile; is a legal statement.

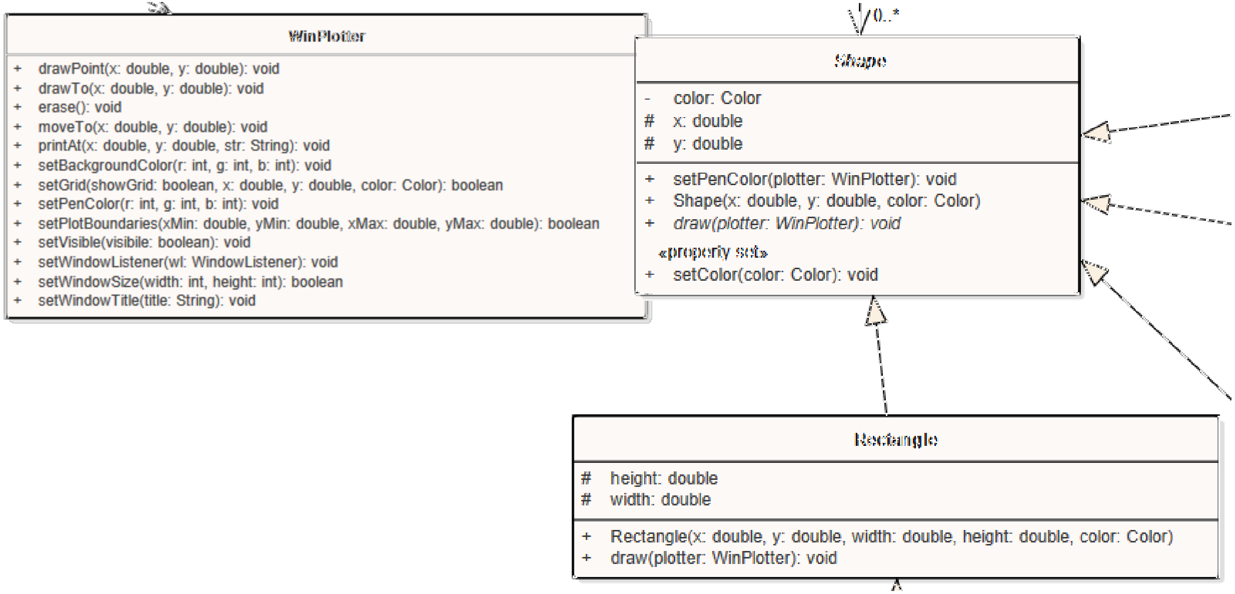
(e) ______ bTile.toString() is a legal statement.

(f) ______ fTile.playerVisits() is a legal statement.

(g) ______ position is an instance variable of the Tile class that must be declared `final`.

5. (15 points) Suppose you are interviewing for an internship position and are asked: “Is inheritance necessary in object oriented programming? If so, give an example of something that could not be done without it. If not, explain why, even though it is not necessary, inheritance is an important feature of the Java language.” Your grade will depend on how well you convince the interviewer that you should be hired.

For the remaining problems, refer to a portion of the lab 3 UML class diagram below:


0..*
WinPlotter
+ drawPoint(x: double, y: double): void
+ drawTo(x: double, y: double): void
+ erase(): void
+ moveTo(x: double, y: double): void
+ printAt(x: double, y: double, str: String): void
+ setBackgroundColor(r: int, g: int, b: int): void
+ setGrid(showGrid: boolean, x: double, y: double, color: Color): boolean
+ setPenColor(r: int, g: int, b: int): void
+ setPlotBoundaries(xMin: double, yMin: double, xMax: double, yMax: double): boolean
+ setVisible(visibile: boolean): void
+ setWindowListener(wl: WindowListener): void
+ setWindowSize(width: int, height: int): boolean
+ setWindowTitle(title: String): void
Shape
- color: Color
# x: double
# y: double
+ setPenColor(plotter: WinPlotter): void
+ Shape(x: double, y: double, color: Color)
+ draw(plotter: WinPlotter): void
«property set»
+ setColor(color: Color): void
Rectangle
# height: double
# width: double
+ Rectangle(x: double, y: double, width: double, height: double, color: Color)
+ draw(plotter: WinPlotter): void

Suppose the design is changed so that the Shape class no longer has a draw() method. Instead, a Drawable interface with one method: `void` draw(WinPlotter plotter) is created, and the Rectangle class implements the Drawable interface.

7. (10 points) Implement the complete Drawable interface described in the previous problem.

8. (15 points) Implement the complete Rectangle class as modified in problem 7.

Additional space — indentify which problem your work is associated with.