Quiz 3
Study Guide / Practice Problems

Topics:

- Interfaces Overview
    - Resources:
        - Module 09
            - Interfaces intro
        - Lab 3
    - You should know:
        - The purpose of interfaces
        - Rules about what can/can't go in an interface
        - How to pick methods that should go in an interface
        - How to loop through a list of interface items and grab specific objects out

- Compiler Rules (Relating to Interfaces)
    - Resources:
        - Module 11
            - Worksheet (CR / A / W)
    - You should know:
        - Rules with upcasting/downcasting
        - How to tell if something will compile, crash on runtime, or not compile
- UML
    - You should know how to represent/read relationships between classes and interfaces/read simple UML
        - Dashed lines vs solid
        - <<interface>> written vs not
        - <<static>> or ____ for static vs not

Study Material:

- lecture material, this practice quiz, lab 3, old quiz material

Practice problems

1. Interfaces

A.

Review lab 3 and be able to:

- identify common methods to put in an interface
- make and implement an interface
- Be able to loop through a list whose reference variable type is an interface and access specific types out of it using instanceof

B.

Assume you have two different interfaces, Paintable and outline. Paitable allows for a shape to be filled in with a specific opacity (that ranges from 0 to 1.0, with 1.0 being completely opaque and 0 being completely transparent) and color. Both interfaces are included below:

```
public interface Paintable {
    void setOpacity(double percent);
    void setColor(Color inC);
    double getOpacity();
}

public interface Outline {
    void setLineWeight(int pixWidth);
    void setLineColor(Color inC);
    double getLineWeight();
}
```

Now also assume you have various classes as follows:

```
public class Circle implements Paintable, Outline
{...}
public class Rectangle implements Paintable {...}
public class Polygon implements Outline {...}
```

And the following instances of these classes:

```
Paintable primitive = new...;
```

```
Outline cartoon = new...;
Circle myCircle = new...;
Rectangle myRectangle = new...;
Polygon myPoly = new...;
```

And that: The class **`Circle`** contains the method:

```
public void intersectWith(Circle otherCircle)
```

The class **`Rectangle`** contains the method:

```
 public void rotate(double radians)
```

The class **`Polygon`** contains the method:

```
public void numberVerts()
```

Now, in the column to the right of each code fragment below, write A if the fragment will always compile and run , CR if the class will compile but might crash at runtime , or WC if it will not compile.

Code Fragment	**A, CR, or WC?**
`myCircle.intersectWith((Circle)myPoly);`	
`primitive = (Paintable)myCircle;`	
`myRectangle.rotate(.707);`	
`cartoon = (Outline)myRectangle;`	
`myCircle = cartoon;`	
`myCircle.getOpacity()`	
`cartoon = new Outline();`	
`((Polygon)cartoon).numberVerts()`	
`primitive = myPoly;`	

`cartoon = new Circle(....);`	