Here's a problem from a previous Final Exam.

Consider the following skeleton for a Robot class, which has private fields for storing the location of a Robot object, its name, and the direction it’s facing (North for a direction parallel to the positive y axis, South for the negative y axis, East for the positive x axis, or West for the negative x axis). It also has stub methods for constructing a Robot object, changing the direction, and moving the location of the robot in the direction it’s facing.

```java
public class Robot {
    private String name;
    private char direction; // 'N', 'S', 'E', or 'W'
    private int xLoc, yLoc; // the (x, y) location of the robot

    // Initialize name, direction, and (x, y) location
    public Robot(String name, char dir, int x, int y) { ... }

    public String toString() {
        return name + " is standing at (" + x + "," + y + ") and facing" + direction);
    }

    // turn 90 degrees clockwise, e.g. 'N' changes to 'E', 'E' to 'S', ...
    public void turnClockwise() { ... }

    // turn 90 degrees counterclockwise, e.g. 'N' to 'W', 'W' to 'S', ...
    public void turnCounterClockwise() { ... }

    // move numSteps in direction you are facing, 
    // e.g. if 'N' 3 steps, then y increases 3
    public void takeSteps(int numSteps) { ... }
}
```
(a) Assuming the class above is completed correctly, what does the following program display on the screen:

```java
public static void main(String args[]) {
    Robot robbie = new Robot("Robby", 'N', 10, 12);
    for (int i = 0; i < 5; i++)
    {
        if (i % 2 == 0)
        {
            robbie.turnClockwise();
        }
        else
        {
            robbie.turnCounterClockwise();
        }
        robbie.takeSteps(3);
        System.out.println(robbie);
    }
}
```

(b) Complete the constructor, the `turnClockwise` method, and the `takeSteps` method. Make sure your constructor validates its input. You do not need to define `turnCounterClockwise`.

(c) Write Java code to create an array of 5 robots. Use a for loop to fill in the array so that the n-th robot is named "robot n", and it starts off life facing east at location (n, n).
class SuperClass
{
    protected int x = 0;

    public SuperClass(int x)
    {
        this.x = x;
    }

    private void increment() { x++; }
    protected final void add(int y)
    {
        x += y;
    }
    public void display()
    {
        System.out.println(x);
    }
}

public class SubClass extends SuperClass
{
    public SubClass(int x)
    {
        super(x);
    }
    public void display()
    {
        add(2);
        super.display();
    }
    public static void main(String [] args)
    {
        SuperClass sc = new SuperClass(3);
        sc.display();

        sc = new SubClass(3);
        sc.display();
    }
}

(a) List the name of all methods that subclasses of SuperClass inherit.
(b) List the name of all methods that are visible in subclasses of SuperClass (in other words, methods that can be called directly).

(c) List the name of all methods that may NOT be overridden by any subclasses of SuperClass.

(d) What gets displayed on the screen when SubClass is executed?
<table>
<thead>
<tr>
<th>Public class Alpha</th>
<th>Public class Beta</th>
<th>Public class Nu</th>
</tr>
</thead>
<tbody>
<tr>
<td>public int a = 0;</td>
<td>protected int a = 0;</td>
<td>public Nu() {</td>
</tr>
<tr>
<td>public void alpha() {</td>
<td>public Beta(int x) {</td>
<td></td>
</tr>
<tr>
<td>a++;</td>
<td>this.a = x;</td>
<td></td>
</tr>
<tr>
<td>System.out.println(a);</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>public void beta() {</td>
<td></td>
</tr>
<tr>
<td>public static void main (String [] args) {</td>
<td>a+=2;</td>
<td></td>
</tr>
<tr>
<td>Alpha a = new Alpha();</td>
<td>System.out.println(a);</td>
<td></td>
</tr>
<tr>
<td>a.alpha();</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>public static void main (String [] args) {</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>Nu n = new Nu();</td>
<td></td>
</tr>
<tr>
<td>Output from executing the Alpha class:</td>
<td>n.nu();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>n.beta();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>n.alpha();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Beta b = new Beta(10);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b.alpha();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b.beta();</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Output from executing the Beta class:</td>
<td>Output from executing the Nu class:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>