AP CS Unit 5: More Classes and Objects Programs

1. Complete the Segment class.	import java.awt.Point;
	
	public class Segment{
	private Point a, b;
	
	public Segment(){
	instantiate two Point objects that have random x and y coordinates between -5 and +5. use Math.random
	}
	
	public double length(){
	returns the distance between the two points
	}
	
	write accessor methods called getA and getB
	}

Test your code by running the RunSegments class.	import java.awt.Point;
	
	public class RunSegments{
	public static void main(String [] args){
	Segment s1 = new Segment();
	Point p1 = s1.getA();
	System.out.println(p1.toString());
	Point p2 = s1.getB();
	System.out.println(p2.toString());
	System.out.println(s1.length());
	
	Segment s2 = new Segment();
	p1 = s2.getA();
	System.out.println(p1.toString());
	p2 = s2.getB();
	System.out.println(p2.toString());
	System.out.println(s2.length());
	}
	}

Here is some sample output (yours will differ because of the random points)

```
java.awt.Point[x=1,y=-1]
java.awt.Point[x=-2,y=-4]
4.242640687119285
java.awt.Point[x=1,y=5]
java.awt.Point[x=1,y=-2]
7.0
```

2. Big objects have only one public method, print, which prints any C, E, F, and/or L letters using asterisks. For example, the following:

```
Big b = new Big( "FEEL" );
b.print();
```

Displays this:

```
******
*
***
*
*

******
*
***
*
******

******
*
***
*
******

*
*
*
*
******
```

Please complete the six helper methods. Then write a runner class to test your solution.

The intent of this exercise, and others, is to demonstrate the usefulness of breaking a method into smaller helper methods. (i.e. functional decomposition).

```
public class Big{
    private String message;

    public Big( String s ){
        message = s.toUpperCase();
    }

    public void print(){
        for ( int k = 0; k < message.length(); k++ ){
            String letter = message.substring(k,k+1);
            if ( letter.equals( "C" ) )
                printC();
            else if ( letter.equals( "E" ) )
                printE();
            else if ( letter.equals( "F" ) )
                printF();
            else if ( letter.equals( "L" ) )
                printL();

            System.out.println();
        }
    }

    private void printC(){
        printHorz(6);
        printVert(3);
        one more line of code
    }

    private void printE(){
        complete using printHorz and printVert
    }

    private void printF(){
        complete using printHorz and printVert
    }

    private void printL(){
        complete using printHorz and printVert
    }

    private void printHorz( int n ){
        prints n asterisks horizontally followed by a line
break
    }

    private void printVert( int n ){
        prints n asterisks vertically
    }
}
```

3. Complete the Name class.	public class Name {
	private String first, middle, last;
	
	public Name(String str) {
	str = remove_spaces(str);
	assign_names(str);
	}
	
	private String remove_spaces(String s){
	This returns a String that is identical to s except that all leading and trailing spaces have been removed. In addition, if there are two or more adjacent spaces then all but one space are removed.
	}
	
	private void assign_names(String nm){
	nm contains a name such as John Jacob Jingleheimer Schmidt or Tuhin Gupta. You need to break nm into first, middle, and last names and correctly assign them to the instance variables. You may assume that nm has at least two names.
	If there are 2 names then the middle should be an empty String.
	If there are 3 names, easy.
	If there are more than 3 names, then the first name goes to first and the last name goes to last. All the other names become the middle name.
	}
	
	public String toString(){
	if (middle.length() > 0)
	return first + " " + middle + " " + last;
	else
	return first + " " + last;
	}
	}

Here is some sample code to test your solutions.

```
Name n1 = new Name( "Ann          Marie          Lopez  " );
String s1 = n1.toString();
System.out.println( s1 );              // Ann Marie Lopez
System.out.println( s1.length() );     // 15

Name n2 = new Name( "     John     Woo   " );
String s2 = n2.toString();
System.out.println( s2 );              // John Woo
System.out.println( s2.length() );     // 8

Name n3 = new Name( "   Robert   Oscar Sam   Edward    Benjamin Ulysses   David   " );
String s3 = n3.toString();
System.out.println( s3 );        // Robert Oscar Sam Edward Benjamin Ulysses David
System.out.println( s3.length() );     // 46
```

4. The Rational class encapsulates the idea of a rational number - a number that is the ratio of two integers. The denominator must be positive.

Complete the Rational class. Here is some client code to test your solutions.

```
Rational r1 = new Rational( -10, 30 );
Rational r2 = new Rational( 1, 2 );
Rational r3 = new Rational( -5, -6 );
Rational r4 = new Rational( 3, -4 );
Rational r5 = new Rational( 7, 3 );

System.out.println( r1.toString() );
                        //    -1/3

Rational r6 = r2.multiply( r3 );
System.out.println( r6.toString() );
                        //   5/12

Rational r7 = r4.multiply( r1 );
System.out.println( r7.toString() );
                        //   1/4
System.out.println( r7.value() );
                        // 0.25

Rational r8 = r2.add( r4 );
System.out.println( r8.toString() );
                        //    -1/4

r1 = r4.add( r1 );
System.out.println( r1.toString() );
                        //   -13/12

r1 = r1.add( r1 );
System.out.println( r1.toString() );
                        //   -13/6

Rational r9 = r5.add( r1 );
System.out.println( r9.toString() );
                        //   1/6
System.out.println( r9.value() );
        // 0.16666666666666666
```

```
public class Rational{
    private int num, denom;

    public Rational( int n, int d ){
        if ( d == 0 )
            throw new IllegalArgumentException(
                        "denominator cannot be zero" );

        if ( d < 0 ){
            n = -1*n;
            d = -1*d;
        }
        num = n;
        denom = d;
        reduce();
    }

    public Rational multiply( Rational r ){
        int  top = num * r.num;
        int bottom = denom * r.denom;
        return  new Rational( top, bottom );
    }

    public Rational add( Rational r ){
        Returns a Rational object that is the sum of
this object and r.  The denominators may be different.
    }

    // this method gets used in problem 5
    public boolean same( Rational r ){
        return num == r.num && denom == r.denom;
    }

    public double value(){
        returns value of the fraction as a double
    }

    private void reduce(){
        Find the greatest common factor between the
numerator and the denomination.  Use this number to
simplify the fraction.
    }

    public String toString(){
        return num + "/" + denom;
    }
}
```

For the purposes of problem 5, the standard form of a linear equation is:

$Ax + By = C$

where:

A and B cannot both be zero.
The leading non-zero coefficient should be positive.
A, B, and C are relatively prime integers (they have no common factors other than 1).

5. Complete the Equation class. It encapsulates the idea of a linear equation in standard form. Code to test your solution is on the next page.

```
public class Equation{
	private int a, b, c;

	public Equation( int a1, int b1, int c1 ){
		if ( a1 == 0  && b1 == 0 )
			throw new IllegalArgumentException( "a and b cannot both be zero" );

		modify the inputs as needed to fit the above definition of standard form
		reduce();
	}

	public boolean same( Equation e ){
		Returns true if this equation is the same as e; otherwise it returns false.
	}

	public Rational slope(){
		Returns the slope of the line as a Rational object.
		Returns null if the slope is undefined.
	}

	public boolean perpendicular( Equation e ){
		Returns true if this equation is perpendicular to e; otherwise it returns false.
	}

	private void reduce(){
		Find the greatest common factor to the coefficients then divide them by it.
	}

	public String toString(){
		String s = "";
		if ( a > 0 ) {
			s = a + "x";
			if ( b > 0 )
				s += " + " + b + "y";
			else if ( b < 0 )
				s += " " + b + "y";
			return s + " = " + c;
		}

		return b + "y = " + c;		// a is zero
	}
}
```

```
public class RunEquation{
	public static void main( String [] args ){
		Equation e1 = new Equation( 2, -3, 12 );
		Equation e2 = new Equation( -8, 12, -48 );
		Equation e3 = new Equation( 0, -7, 6 );
		Equation e4 = new Equation( 10, 0, 15 );
		Equation e5 = new Equation( 18, 12, 12 );
		Equation e6 = new Equation( -6, 9, 3 );

		System.out.println( e1.toString() );		//	2x -3y = 12
		System.out.println( e2.toString() );		//	2x -3y = 12
		System.out.println( e3.toString() );		//	7y = -6
		System.out.println( e4.toString() );		//	2x = 3
		System.out.println( e5.toString() );		//	3x + 2y = 2
		System.out.println( e6.toString() );		//	2x -3y = -1

		System.out.println( e1.same(e2) );		//	true
		System.out.println( e1.same(e5) );		//	false
		System.out.println( e3.same(e3) );		//	true

		Rational r = e1.slope();
		if ( r != null )
			System.out.println( r.toString() );	// 2/3
		else
			System.out.println( r );

		r = e4.slope();
		if ( r != null )
			System.out.println( r.toString() );
		else
			System.out.println( r );		// null

		r = e6.slope();
		if ( r != null )
			System.out.println( r.toString() );
		else
			System.out.println( r );		// 2/3

		System.out.println( e1.perpendicular( e3 ) );	//	false
		System.out.println( e3.perpendicular( e4 ) );	//	true
		System.out.println( e5.perpendicular( e1 ) );	//	true
		System.out.println( e4.perpendicular( e2 ) );	//	false
	}
}
```