Problem 1.2.2

```
/*
*   Problem 1.2.2
*
*   Author: Alexander Winkles
*
*   Purpose: This code computes the value of sin^2(theta)+sin^2(theta) and
*               returns its difference from 1 to confirm that Java approximates
*               trigonometric functions properly.
*
*   Compile: javac Problem2.java
*
*   Input: A command line argument of type double.
*
*   Output: A double value representing the difference of sin^2(theta)+sin^2(theta)
*           and 1.
*
*   Run: java Problem2 args[0]
*
*   Examples:
*       1. 3.1415 --> 0.0
*       2. 278 --> -1.1102e-16
*/

public class Problem2 {
    public static void main(String[] args) {

        // Reads in the command line value and assigns it to the variable theta

        double theta = Double.parseDouble(args[0]);

        // Performs a comparison of sin^2(theta)+cos^2(theta) and 1.

        double result = Math.sin(theta)*Math.sin(theta) + Math.cos(theta)*Math.cos(theta) - 1.0;

        System.out.println("1 - sin^2(theta)+cos^2(theta) = " + result);
```

```
    }
}
```

These values are not always exactly 1 because $\sin^2\theta$ and $\cos^2\theta$ are decimal approximations, so precision is lost.

Problem 1.2.7

These will print out the following:

(a) 2bc

(b) 5bc

(c) 5bc

(d) bc5

(e) bc23

This can be understood by realizing that within System.out.println, integers are converted to strings that are concatenated to other strings. However, there is an order of operations in place that allows 3 and 2 to be added before converted.

```
/*
* Problem 1.2.7
*
* Author: Alexander Winkles
*
* Purpose: This program is designed to check the ways System.out.println
*           converts types of data depending on the order they are presented.
*
* Compile: javac Problem7.java
*
* Input: None
*
* Output: Five print statements
*
* Run: java Problem7
*
*/
public class Problem7 {
```

```
    public static void main(String[] args){
        System.out.println(2 + "bc");
        System.out.println(2+3+"bc");
        System.out.println((2+3)+"bc");
        System.out.println("bc"+(2+3));
        System.out.println("bc"+2+3);
    }
}
```

Problem 1.2.13

The result of this statement is false, because Math.sqrt(2) is only an approximation, so squaring it will not return 2 exactly.

```
/*
* Problem 1.2.13
*
* Author: Alexander Winkles
*
* Purpose: To demonstrate that methods such as Math.sqrt() return approximate
*           answers rather than exact values.
*
* Complie: javac Problem13.java
*
* Input: None
*
* Output: A print statement
*
* Run: java Problem13
*
*/
public class Problem13 {
    public static void main(String[] args) {

        boolean result = (Math.sqrt(2)*Math.sqrt(2) == 2);

        System.out.println("The result of Math.sqrt(2)*Math.sqrt(2) == 2 is " + result);
```

```
    }
}
```

Problem 1.2.19

```
/*
* Problem 1.2.19
*
* Author: Alexander Winkles
*
* Purpose: This program takes two integers from the command line and returns a
*           random interger between them.
*
* Complie: javac Problem19.java
*
* Input: int a, b
*
* Output: An integer between a and b
*
* Run: java Problem19
*
*/

public class Problem19 {
    public static void main(String[] args) {

        // takes integers from the command line and assigns them to a and b

        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);

        // creates a new variable that generates a random double between a and b

        double randomGenerator = (b-a)*Math.random() + a;

        // rounds the result from above to an integer
```

```
        long result = Math.round(randomGenerator);

        System.out.println(result);

    }
}
```

Problem 1.2.26

```
/*
* Problem 1.2.26
*
* Author: Alexander Winkles
*
* Purpose: This program converts Cartesian to polar coordinates.
*
* Complie: javac Problem26.java
*
* Input: Takes two real numbers x and y (Cartesian coords.) from the command line
*
* Output: Returns the polar coordinates corresponding to inputs
*
* Run: java Problem26
*
* Examples:
*   1. (1.0, 0.0) --> (1.0, 3.1415)
*   2. (0.0, 1.0) --> (1.0, 1.57079)
*
*/

public class Problem26 {
    public static void main(String[] args) {

        // assigns values from the command line to x and y

        double x = Double.parseDouble(args[0]);
```

```
        double y = Double.parseDouble(args[1]);

        // computes r and theta, the polar coordinates

        double r = Math.sqrt(x*x + y*y);
        double theta = Math.atan2(y,x);

        // adjusts theta to be in the range [0,2*PI] rather than [-PI,PI]

        if (theta < 0) {
            theta += 2*Math.PI;
        }

        // returns results nicely

        System.out.println("(" + x + ", " + y + ") in polor coordinates are (" + r + ", " + theta + ").");

    }
}
```

Problem 1.2.27 (Honors Option)

```
/*
* Problem 1.2.27
*
* Author: Alexander Winkles
*
* Purpose: This program generatesa random number from the
*           Gaussian distribution based on the Box-Muller formula.
*
* Complie: javac Problem27.java
*
* Input: None
*
* Output: A random number w.
*
* Run: java Problem27
```

```
*
*/

public class Problem27 {
    public static void main(String[] args) {

        // Generates random numbers between 0 and 1

        double u = Math.random();
        double v = Math.random();

        // Computes a random number using Box-Muller

        double w = Math.sin(2*Math.PI*v)*Math.sqrt(-2*Math.log(u));

        System.out.println(w);

    }
}
```
