__EXERCISE SOLUTIONS__

1. Write a `while` loop that prints the numbers from 1 to 20, as well as their squares, in this format:

```
1 squared = 1
2 squared = 4
3 squared = 9
```

```
int i = 1;
while (i <= 20)
{
    System.out.println(i + " squared = " + i*i);
    i++;
}
```

2. Write a `for` loop that counts from 0 to 100 and prints out each number.

```
for (int i = 0; i <= 100; i++)
{
    System.out.println(i);      // curly braces are optional here
}
```

3. Write a `while` loop that asks the user to enter a series of positive numbers that will be added. The loop stops accepting input when the user enters a 0. Then print out the sum of those numbers.

```
Scanner in = new Scanner(System.in);
double sum = 0;
double input = 0;
while (input != 0)
{
    System.out.print("Enter a positive number to be added (0 to quit): ");
    input = in.nextDouble();
    if (input > 0)
        sum += input;
    else if (input < 0)
        System.out.println("Negative numbers aren't allowed");
}
System.out.println("The sum of the positive numbers you entered is: " + sum);
```

4. Write a `for` loop that prints out the numbers 1, 4, 7, 10, 13, ... , 298, 301.

```
for (int i = 1; i <= 301; i = i + 3)
    System.out.println(i);
```

5. Write a `while` loop that prints out the numbers 0, 4, 8, 12, ... , 96, 100.

```
int i = 0;
while (i <= 100)
{
    System.out.println(i);
    i += 4;
}
```

6. Write an infinite loop that has the user repeatedly enter passwords until he/she enters the correct password, a password of your choosing. Once the password is entered, `break` out of the infinite loop.

```
Scanner in = new Scanner(System.in);
String thePassword = "70p53cr37";
String input = "";
while(true)
{
    System.out.print("Password: ");
    input = in.nextLine();
    if (input.equals(thePassword))
        break;
    System.out.println("Invalid entry.");
}
// If we get here, they've entered the correct password!
```

7. Write a loop that displays the Fibonacci sequence. The first two numbers in the Fibonacci sequence are `0` and `1`. Subsequent numbers are found by adding the previous two numbers, so the sequence begins `0, 1, 1, 2, 3, 5, 8, 13, ...`

There are ways to solve this using something called *recursion*, but here we'll just use a loop to solve it. The problem doesn't indicate how many times we should run the loop, and the Fibonacci sequence is infinite, so... I guess this will be an infinite loop that the user will have to manually break out of.

```
int firstNum = 0;
int secondNum = 1;
System.out.println("The Fibonacci series (Ctrl-C to quit)");
while(true)        // infinite loop!
{
    System.out.println(firstNum);
    int temp = firstNum + secondNum;
    firstNum = secondNum;
    secondNum = temp;
}
```

8. Write a "prime finder" loop that determines whether a given number `n` is prime or not. Any integer `n > 2` is prime if no number between `2` and $\sqrt{n}$ (inclusive) evenly divides into `n`. The loop should return `true` if `n` is prime and `false` if `n` is not prime.

```
for (int i = 2; i <= Math.sqrt(n); i++)
{
    // Each time we go through the loop, check to see if the current i divides
    // evenly into the prospective prime n. If it does, we know this number is
    // not prime so we should return false.

    if (n % i == 0) return false;  // it's not a prime
}

// If we fall out of the loop having not returned false, it must be that none
// of those numbers divided evenly into our prospective prime. Therefore, it
// must be prime and we can return true.

return true;  // it IS a prime!
```