Gentry Atkinson
CS5318- Spring 2019
Homework #1
Due: 19 Feb 2019

1. Fill the following blanks:
    1. When two different methods have the same name in a class, this is an example of **overloading**.
    2. A (An) **interface** is like a class except that it contains only instance methods, no instance variables.
    3. Instance variables and instance methods that are declared **public** or **protected** are inherited by the subclasses.
    4. A method that is invoked when an object is created is known as a **constructor**.
    5. Constants should be declared **final**.
2. If x is an object of class A and y is an object of class B that is a subclass of A, after the assignment x = y, why is x.m() illegal when m is a method of B but not A? **Although x and y now reference the same object, the compiler still sees x as being of type A and so only methods available to A are legal calls for x.**
3. Write a Java program segment that prints the index of the first all-zero row of an nXn integer matrix M. The code should access each element of the matrix at most once and should not access rows beyond the first all-zero row and columns within any row beyond the first non-zero element. It should have no variables except the matrix M and two loop indices row and column.

```
int row, column;

for (row = 0; row < M.length - 1; ++row) {

        for (col = 0; col < M[row].length - 1; ++ col {

                if (M[row][col] != 0) break;

        }

        if (col == (M[row].length -1) {

                System.out.println ("First all zero row is: " + row);

                break;

        }

}
```

4. Java has the >>> operator, but C does not. What does the >>> do? Why is it needed in Java but not in C?

    The right bit shift operator in C did not maintain the sign bit of the leftmost bit. The >> operator in Java is a signed shift operator and always maintains the sign of negative integer values. The >>> operator in Java does not maintain the sign bit and functions more like the legacy C style >> operator.

5. Why is it not possible to create an object of an abstract class? Does this mean that variables of abstract class types cannot be declared? Explain.

    Abstract classes do not have fully implemented methods and so objects cannot be instantiated from abstract classes. However, it is still possible to declare a variable that references an abstract class. This will allow that variable to call any legal methods on any child class of the parent, abstract type.