CSE 220 Homework Assignment 2 (Due 6/18/24)

1) (40 pts) This question deals with concepts related to *classes* and *encapsulation*. Your answers to each part should be 2-5 sentences apiece.

a. Explain the difference between an instance variable and local variable in Java.

b. Explain the impact of making an instance variable private in Java, and why this is commonly done.

When a variable is made private in Java, it is only accessible by member functions of the class it is in. This is done so that it can only be modified by methods of that class, on that instance.

c. It is common for the keywords **private**, **static**, and **final** to be confused by someone new to Java. Explain the principal differences between i. making a variable private, ii. making a variable static, and iii. making a variable final.

Private, as discussed above, is an encapsulation method used to ensure that a variable is only modifiable inside the scope of the class in which it is defined.

Static variables belong to the class, but not any instance of that class. There is only one copy of that variable for all instances of the class.

A **final** is the same thing as a "const" in C and C++. The value, once assigned to this variable, cannot be modified. In other words, finals are immutable.

NOTE: these behaviors can be combined in variables. For instance, a **private static final** variable is allowed and would be only accessible by methods of that class, would be a single copy shared across instances of the class, and would be constant.

d. Explain what the term **constructor** means in the context of Java, and explain what it means to have a constructor **overloaded**.

As is the case in many other languages, constructors create class instances. They are special functions with the same name as the class, and no return. When you create an instance of an object, the constructor for that object is called.

Constructor overloading allows the creation of multiple constructor functions for a single class. This can enable flexible instantiation of class objects, because you can create more constructors with different parameter lists or initial values.