AP Computer Science
Object Oriented Terminology and Concepts

Terms to know:

- **Accessor Methods:** Methods that allow client code to look at a class's instance variables without giving that client code the ability to modify those variables.
- **Constructor:** Creates the object and assigns initial values to the instance variables. It is a method that must be called the same name as the class, and has no return type.
- **Encapsulation:** A design goal to help guide correct design of classes. The main idea is that you want to hide the data (instance variables) behind a wall, and only allow client code to interact with that data through predefined means (public methods). Classes must therefore be self-sufficient so that all situations can be handled behind this wall. A properly encapsulated class makes unbreakable objects. For example: denominators in Fraction class were impossible to set to zero. It also makes it easier to interact with these objects because you only have to know how to interact with them, not how they work (Information Hiding).
- **Instance:** Another word for object. The line of code: Fraction f = new Fraction("1/3"); creates an instance of the Fraction class. The variable f is a pointer to that instance.
- **Instance Variables:** Variables declared within a class between the class header and the Constructor. All instances of this class will have their own independent copies of these variables.
- **Instantiate:** To create a new instance of a class, accomplished with the "new" keyword and the Constructor.
- **Mutator Methods:** Methods that allow client code to change an object's instance variables.
- **The "new" keyword:** A keyword that you use to invoke a Constructor to create a new instance of that class. (Strings are an exception).
- **Overloaded Methods:** Methods that share the same name but different parameters. For example, we overloaded the constructor in the Fraction class. Overloading keeps it easy for client programmers to interact with your code.
- **Private:** Variables and methods declared private can only be directly accessed from within that class (not even subclasses).
- **Public:** Variables and methods declared public can be directly accessed from anywhere that class or objects of that class are accessible.
- **Static:** Static means it is connected to the class, not to objects of that class. If a class has a static variable, objects do not get their own copy of this variable; ALL instances of this class share the same value for static variable. For example, a Circle class would have non-static variables to represent things like radius, x and y coordinates, and other things that are unique to each instance of that class, but they might have a static variable for PI, since all instance of this class have the same value of Pi.

  Static methods are not called from objects, but rather the class. For example, for a class called Foo that has a static method bar() and a non-static method "baz()":

Calling Static Method: Foo.bar();

Non-Static Method:
```
Foo foo = new Foo();
foo.baz();
```

- **Client Code:** In relation to a given class, client code refers to any code that is OUTSIDE of this class.

1. What is the difference between a class and an object?

Class:	Object:
A piece of the program's source code	An entity in a running program
Written by a programmer	Created when the program is running
Specifies the structure (the number and types of attributes) for the objects of this class, the same for all of its objects	Holds specific values of attributes; some of these values can change while the program is running
Specifies the possible behaviors of its objects — the same for all of its objects	Behaves appropriately when called upon
A Java program's source code consists of several classes	A running program can create any number of objects (instances) of a class
Like a blueprint for building cars of a particular model	Like a car of a particular model that you can drive

2. Fill in the blanks:

QUESTION	CLASS	OBJECT
Where does it exist?	Written into the source code	The computer's running memory
Who created it?	The programmer	Created by the program when it is running
What is its purpose?	A blueprint that specifies the state and behaviours of the objects of this type that will be built.	An instance of the class that behaves as designed when called upon.

3. What is the advantage of making your instance variables private? Private variables can only be accessed through predefined public methods that can filter out any shenanigans that might

cause unexpected behaviour or fatal errors (for example: setting the denominator to zero in the Fraction class).

4. Explain the difference between a static method and a non-static method. Static methods are not called from objects, but rather the class. For example, for a class called Foo that has a static method bar() and a non-static method "baz()":

Calling Static Method:	Foo.bar();
Non-Static Method:	Foo foo = new Foo(); foo.baz();

5. Explain the difference between a static variable and non-static variable. If a class has a static variable, objects do not get their own copy of this variable; ALL instances of this class share the same value for static variable. For example, a Circle class would have non-static variables to represent things like radius, x and y coords, and other things that are unique to each instance of that class, but they might have a static variable for PI, since all instance of this class have the same value of Pi.