Basic Java Class Syntax

Important: A class called *ClassName* must be stored in a file called *ClassName*.java.

```
public class ClassName {

  // "Fields" or "instance variables":

  public/private dataType variableName;
  public/private dataType variableName2;
  public/private dataType variableName3;  // etc...

  // "Instance methods" (functions that can be called on these objects):
  public/private returnDataType methodName(type1 param1, type2 param2, ...) {
    // code goes here
  }

  public/private returnDataType methodName2(type1 param1, type2 param2, ...) {
    // code goes here
  }
}
```

- The state of a class is represented through defining *instance variables* or *fields*. These are variables defined inside the class, and *every object of that class has its own copy of these variables*.

- The behavior of a class is represented through defining *instance methods*. These are functions defined inside the class. The code in an instance method may refer to the instance variables defined in the class. When an instance method is called on an object, that object's values for the instance variables are used when the code refers to an instance variable.

- Inside a class definition, instance variables and instance methods may be referred to by their normal names, without the "dot" prefix. Outside of a class, however, you must use the "dot" syntax to refer to these variables or call these methods.

Definitions Review:

- **Class**: A template or blueprint for creating objects. The code for a class defines the state (instance variables) and behavior (methods) that each instance of that object will have. A class is a data type.

- **Object**: A particular "value" of the class, separate from all other objects of that class. Also: "instance of a class."

- **Reference** (to an object): A data type in Java that "refers to" or "points to" a particular object in memory. All objects in Java are accessed through references.

- **Declaration**: Creating a new variable by associating a data type with a variable name. Ex., **Scanner name;**

- **Instantiation**: Creating a new object with the keyword `new`. Ex., **new Scanner(system.in);**

- **Methods** of a class (particularly "instance methods"): Functions that can be called ("invoked") on objects of that class.

- **Constructor**: One or more special methods of a class that are called when an object is **instantiated**. Usually responsible for initializing the instance variables.