**Constructing an Object**

Objects have attributes and behaviors. Attributes are nouns describing features of an object. Programmers call this a *has a* relationship (e.g. a dog *has a* name, a car *has a* color). Behaviors are verbs describing the actions that objects can perform.

You have probably heard the word default used in the context of defeat or neglect, as in "We lost by default because we did not have enough players" or "The mayor won by default because no one ran against him." But more generally, a default is something that automatically happens. When a numeric or **String** array is declared and initialized, elements are initially set to 0 and null, respectively. Have you gotten the "Identifier may not have been initialized" error? The compiler is complaining because a variable was not assigned an initial default value. Of course, you are not stuck with a default value; a variable's assigned value can be changed by the user or the program (unless it is a constant). In this lesson, you will learn about a very useful default situation in object-oriented programming.

The reserved word **new** in each of the following statements is responsible for invoking a constructor of the respective class, to create a new object.

Scanner in = new Scanner(System.in);  
Scanner inFile = new Scanner(new File("data.txt"));  
PrintWriter outFile = new PrintWriter ("hello.txt");

Every class has at least one constructor, but where is it, and what does it look like? You can see what a constructor looks like simply by going to the [**Java API**](javascript:void(0);) and selecting any class (not an interface) from the list. Immediately before the Method Summary table, you will find each class's Constructor Summary. There will always be at least one constructor listed for each class, the [**default constructor**](javascript:void(0);).

If every class has at least one constructor, and each program you have written represents a class, why haven't you written any constructors in your programs? If you do not supply a constructor, Java automatically provides a default constructor when the class is initially created. The default constructor is often devoid of any code, but it must exist unless you provide your own constructor.

### Part 1

In this portion of the lesson, you will follow the transition of one program through two stages of design: procedural, and object-oriented design. Up until now, all programming took place in the **main ()** method. The procedural to object-oriented design progression is very important, so it is essential that you not rush through it.

* Create a new project called 05.05 Constructors in the Mod05 Lessons folder.
* Download the following programs to the newly-created folder.
  + [**PlanetV2.java**](https://lti.flvsgl.com/flvs-cat-content/s7t7o6n63q14m0ael4kf9g97uo/flvs-cat-session/apcomputersciencea_v20/module05/lesson05/docs/05_05b/planetv2.java)
  + [**PlanetV3.java**](https://lti.flvsgl.com/flvs-cat-content/s7t7o6n63q14m0ael4kf9g97uo/flvs-cat-session/apcomputersciencea_v20/module05/lesson05/docs/05_05b/planetv3.java)
* Open the [**05.05 Virtual Lecture Notes (Part 1)**](https://lti.flvsgl.com/flvs-cat-content/s7t7o6n63q14m0ael4kf9g97uo/flvs-cat-session/apcomputersciencea_v20/module05/lesson05/pop/05_05b/05_05b_pop01.htm).
* As you run each program, read the accompanying description in the Virtual Lecture Notes.

After you feel comfortable with default constructors, modify the **PlanetV3** program to print the name and calculate the radius of other planets.

### Part 2

As the size and complexity of programs grow, it is easy to become overwhelmed by all the classes, objects, methods, variables, parameters, and constructors in an object-oriented program. You need some way to organize all the pieces and parts into a coherent whole. The [**Unified Modeling Language**](javascript:void(0);) (UML) is a standardized organizational design tool used by programmers. Although the topic of the UML is way beyond the scope of this course, there is one aspect that can turn confusion into comprehension: the [**class diagram**](javascript:void(0);). Use this simple graphic organizer as a pre-programming planning tool for all your assignments and you will save enormous amounts of time.

* Open the [**05.05 Virtual Lecture Notes (Part 2)**](https://lti.flvsgl.com/flvs-cat-content/s7t7o6n63q14m0ael4kf9g97uo/flvs-cat-session/apcomputersciencea_v20/module05/lesson05/pop/05_05b/05_05b_pop02.htm).

Inexperienced programmers usually follow the ready-fire-aim approach to writing code. They rush to begin coding without a plan, and then constantly have to adjust their approach as unforeseen issues arise. Never use a contorted design to cover up for poor planning. Aimlessly writing code before you are ready is a waste of time. Even five minutes spent developing a class diagram, **before you start coding**, can save you an hour of frustration. But you may have to learn that the hard way.

**Plan**