**Object Categories**

Name five things you see in your room. List four things you can wear on your feet. Name three things that are not balls but can be thrown in a game.

Everything you listed in each category is an object. Your brain is trained to identify and categorize objects, a natural ability that will serve you well in your development as an object-oriented programmer! Objects with physical characteristics (like dice, darts, and horseshoes) are easy to recognize. Can you name anything with physical characteristics that is **not** an object?

Which of the following is not an object?

smoke  
  
an enchilada  
  
CO2

a bank account  
  
3  
  
BlueJ Shortcut Icon

Pi Symbol  
  
gasoline  
  
a song

If you picked the number 3, you are correct! All the others are objects that can be described using various characteristics.

## The Object in OOP

One of the human brain's most powerful capabilities is pattern recognition. Your brain is hardwired to categorize what you see, hear, smell, taste, touch, and think. Try this experiment to prove it to yourself. How many different ways can you categorize the clothes in your closet?

* Season: winter, spring, summer, fall
* Cost: expensive, reasonably priced, cheap, free, hand-me-downs
* Fabric: cotton, wool, linen, argyle, silk, polyester, denim, etc.
* Activity: formal, school, leisure, sports, etc.
* Apparel: hats, scarves, shirts, blouses, pants, skirts, dresses, socks, shoes, etc.

Each category is logical, but depending on the way your brain is programmed to recognize patterns, you may see additional classification schemes.

Categorizing helps you see the big picture by recognizing relationships and putting things into classes, so use this natural ability to help you master the principles of object-oriented programming.

Before you learn about object-oriented programming (OOP), it is important to distinguish OOP from the procedural programming style.

### Procedural Programming

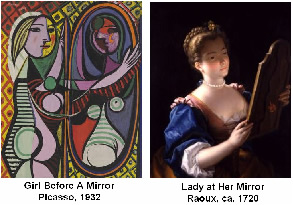
* An approach to software design that centers around procedures.
* A style of programming that decomposes a program into a set of procedures or methods.

### Object-Oriented Programming

* Programs are designed to model real-world or virtual objects that can interact with each other.
* A class acts as a blueprint to define an object's attributes and behaviors.

### Part 1

The distinction between object-oriented programming (OOP) and procedural programming may escape you now, but the sharp contrast is easy to envision, as the following examples illustrate:



* Windows vs. MS DOS
* Lightning vs. a lightning bug
* Picaso vs. Raoux

Programs can certainly be written with either style, but the choice has profound design implications, as you will soon see.

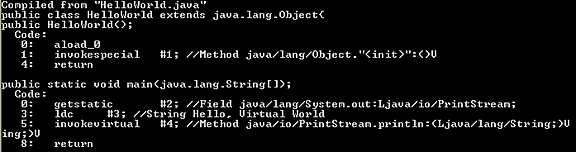
Java's strength is in modeling not only real-world objects, but also in what can be imagined in the virtual world.

Before delving into OOP code, it is a good time to pause and learn a little bit more about what makes Java so different from other programming languages.

### Part 2

Programmers write source code in [**high-level languages**](javascript:void(0);) like Java and C++; however, the computer's [**CPU**](javascript:void(0);) cannot execute these instructions until they are compiled into machine-readable object code. Ideally, a program should run on different computers, but the machine language of different brands of CPU is different, and different brands of computers run different operating systems. Consequently, portability has been a longstanding issue that required developing platform-specific compilers, a time-consuming and expensive process.

In contrast, Java's developers took a different approach to gain platform independence: the [**Java Virtual Machine**](javascript:void(0);) (JVM). The JVM is software written for each platform that acts like a virtual computer superimposed a level above each CPU's machine language. When Java source code is compiled, it produces [**bytecode**](javascript:void(0);), which the Virtual Machine executes and translates into executable object code for a specific CPU. Therefore, a Java program written for a PC running Microsoft Windows may run unaltered on an OSX Macintosh because a JVM exists for both platforms and serves as an interface between the programmer's source code and the CPU's object code. In case you are curious, the following is an example of the bytecode for a HelloWorld program. That is, the Java Virtual Machine is an interpreter for bytecode.



Clearly, it is less time consuming and less expensive to write one JVM for each CPU than to write separate versions of every program for each platform. Java applets that are downloaded over the Internet and run in real time on web pages demonstrate Java's portability and platform independence.