# Overloading

Overloading is when the same method or operator can be used on many different types of data. For instance the + sign is used to add ints as well as concatenate strings. The plus sign behaves differently depending on the type of its arguments. Therefore the plus sign is inherently overloaded.

Methods can be overloaded as well. System.out.println() can print a double, a float, an int, a long, or a String. You don't do anything different depending on the type of number you want the value of. Overloading takes care of it.

Programmer-defined classes can overload methods as well. To do this simply write two methods with the same name but different argument lists. For instance last week you saw several different versions of the Car constructor, one that took three arguments and one that took two arguments, and one that took no arguments. You can use all of these in a single class, though here I only use two because there really aren't any good default values for licensePlate and maxSpeed. On the other hand, 0 is a perfectly reasonable default value for speed.

public class Car {  
  
 private String licensePlate; // e.g. "New York A456 324"  
 private double speed; // kilometers per hour  
 private double maxSpeed; // kilometers per hour  
   
 // constructors  
 public Car(String licensePlate, double maxSpeed) {  
  
 this.licensePlate = licensePlate;   
 this.speed = 0.0;  
 if (maxSpeed >= 0.0) {  
 this.maxSpeed = maxSpeed;  
 }  
 else {  
 maxSpeed = 0.0;  
 }  
   
 }  
  
 public Car(String licensePlate, double speed, double maxSpeed) {  
  
 this.licensePlate = licensePlate;   
 if (maxSpeed >= 0.0) {  
 this.maxSpeed = maxSpeed;  
 }  
 else {  
 maxSpeed = 0.0;  
 }  
   
 if (speed < 0.0) {  
 speed = 0.0;  
 }  
   
 if (speed <= maxSpeed) {  
 this.speed = speed;  
 }  
 else {  
 this.speed = maxSpeed;  
 }  
   
 }  
  
 // other methods...  
   
}

Normally a single identifier refers to exactly one method or constructor. When as above, one identifier refers to more than one method or constructor, the method is said to be overloaded. You could argue that this should be called identifier overloading rather than method overloading since it's the identifier that refers to more than one method, not the method that refers to more than one identifier. However in common usage this is called method overloading.

Which method an identifier refers to depends on the signature. The signature is the number, type, and order of the arguments passed to a method. The signature of the first constructor in the above program is Car(String, double). The signature of the second method is Car(String, double, double). Thus the first version of the Car() constructor is called when there is one String argument followed by one double argument and the second version is used when there is one String argument followed by two double arguments.

If there are no arguments to the constructor, or two or three arguments that aren't the right type in the right order, then the compiler generates an error because it doesn't have a method whose signature matches the requested method call. For example

Error: Method Car(double) not found in class Car.  
Car.java line 17

[Next](http://docs.google.com/02.html) | [Top](http://docs.google.com/index.html) | [Cafe au Lait](http://www.cafeaulait.org/)

Copyright 1997, 1998 Elliotte Rusty Harold

[elharo@metalab.unc.edu](mailto:elharo@metalab.unc.edu)

Last Modified October 20, 1997