Special Topic 8.1

Call by Value and Call by Reference

In Java, parameter variables are initialized with the values that are supplied in the method call when a method starts. Computer scientists refer to this call mechanism as "call by value". There are some limitations to the "call by value" mechanism. As you saw in Common Error 8.1 on page 334, it is not possible to implement methods that modify the contents of number variables. Other programming languages such as C++ support an alternate mechanism, called "call by reference". For example, in C++ it would be an easy matter to write a method that modifies a number, by using a so-called *reference parameter.* Here is the C++ code, for those of you who know C++:

```
// This is C++
class BankAccount
{
public:
   void transfer(double amount, double& otherBalance)
      // otherBalance is a double&, a reference to a double
   {
      balance = balance - amount;
      otherBalance = otherBalance + amount; // Works in C++
   }
   . . .
};
```

You will sometimes read in Java books that "numbers are passed by value, objects are passed by reference". That is technically not quite correct. In Java, objects themselves are never passed as parameters; instead, both numbers and *object references* are passed by value. To see this clearly, let us consider another scenario. This method tries to set the otherAccount parameter to a new object:

```
public class BankAccount
{
   public void transfer(double amount, BankAccount otherAccount)
   {
      balance = balance - amount;
      double newBalance = otherAccount.balance + amount;
      otherAccount = new BankAccount(newBalance); // Won't work
   }
}
```

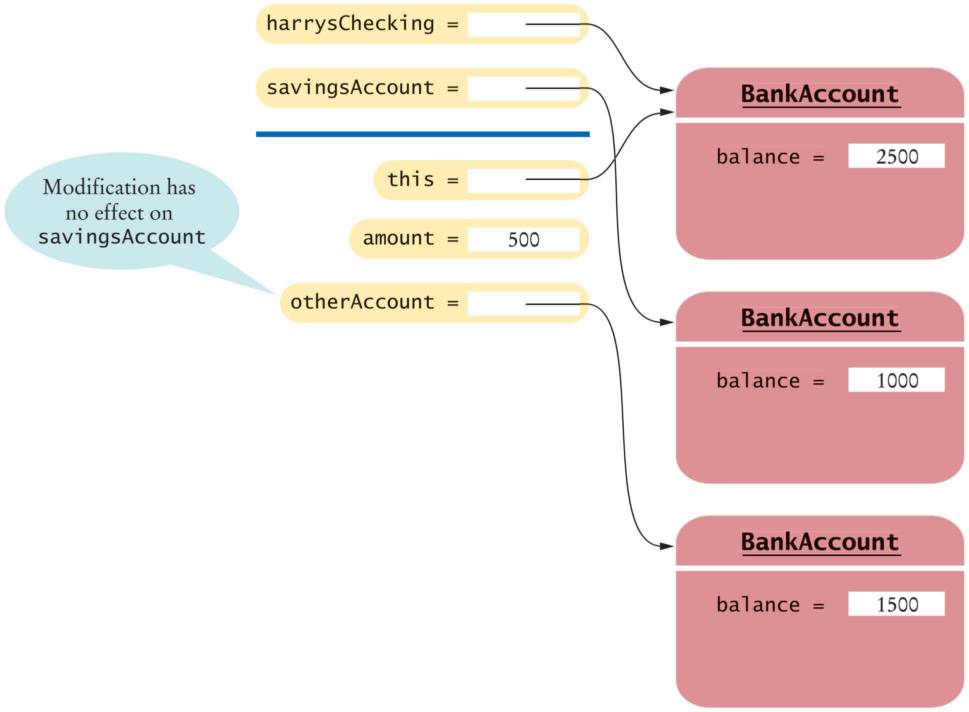
In Java, a method can change the state of an object reference parameter, but it cannot replace the object reference with another.

In this situation, we are not trying to change the state of the object to which the parameter variable otherAccount refers; instead, we are trying to replace the object with a different one (see the figure on page 338). Now the parameter variable otherAccount is replaced with a reference to a new account. But if you call the method with

```
harrysChecking.transfer(500, savingsAccount);
```

then that change does not affect the savingsAccount variable that is supplied in the call.

As you can see, a Java method can update an object's state, but it cannot *replace* the contents of an object reference. This shows that object references are passed by value in Java.



Modifying an Object Reference Parameter Has No Effect on the Caller