11. Inheritance :: Method Signatures and Overloading

An **overloaded** method is one that has the same name as another method but a different **method signature**:

```
public void aMethod(int pX) {
   ...
}
public void aMethod(double pY) {  // aMethod() is overloaded
   ...
}
public void aMethod() {  // aMethod() is overloaded again
   ...
}
```

The signature of a method consists of the **method name** and the **data types (and order)** of the parameters. I use this notation to represent the method signature: <**method: data types**> so the signatures of these three methods are:

```
<aMethod: int>
<aMethod : double>
<aMethod :>
```

11. Inheritance :: Overloading versus Overriding

Since the signatures of overloaded must be different, the compiler can distinguish among them by comparing the data types of the parameters in a method call with the various signatures:

```
someObject.aMethod(4);      // Calls <aMethod : int> since 4 is an int.
someObject.aMethod(3.14);   // Calls <aMethod : double> since 3.14 is a double.
someObject.aMethod();       // Calls <aMethod :> since there are no parameters.
```

Overloaded and overridden methods are two different creatures. An **overridden method** is one that appears in a subclass and does something different than the inherited method from the superclass. An **overloaded** method is one—either in the same class or in a subclass—that has essentially the same behavior as the other methods with the same name, but it does it with different parameter data types.

Overriding always occurs in the context of an inheritance relationship: **a subclass overrides a method inherited from a superclass**. It is not possible for a method in class C to override another method, also in class C.

11. Inheritance :: Overloading/Overriding Example

Overloading occurs either in the context of an inheritance relationship or within a class. Class *Sub* overloads a method in *Super* if those two methods have the same name **but different parameter data types**. Alternative, class *C* may overload a method, also in class *C*, if there are two implementations of a method with the same name, but with different parameter data types.

```
public class Super {
  public void someMethod() { ... }        // someMethod() is overridden in Sub.
  public void aMethod(int pX) { ... }     // aMethod() is overloaded in Super.
  public void aMethod(double pX) { ... }  // aMethod() is overloaded in Super.
}

public class Sub extends Super {
  public void someMethod() { ... }       // someMethod() is overridden in Sub.
  public void aMethod(int pX) { ... }    // aMethod() is overridden in Sub.
  public void aMethod(String pX) { ... } // aMethod() is overloaded in Sub.
}
```

Here, *aMethod*() is **overloaded** in *Super* because there are two implementations of *aMethod*() with different signatures: <aMethod : int> and <aMethod : double>. This example illustrates that **overloading may occur among methods in the same class**.

someMethod() is **overridden** in *Sub* because there is a method in the subclass with the same signature as in the superclass *Super*: <*Super*.aMethod :> and <*Sub*.aMethod:>. This example illustrates that **overriding only occurs in an inheritance relationship**.

11. Inheritance :: Overloading/Overriding Example (continued)

aMethod(**int**) is **overridden** in *Sub* because there is a method in the subclass with the same signature as in the superclass *Super*: <*Super*.aMethod: int> and <*Sub*.aMethod: int>. This example also illustrates that overriding only occurs in an inheritance relationship.

aMethod(*String*) is **overloaded** in *Sub* because there is no other method in *Super* or *Sub* with this signature: <*Sub*.aMethod : String>. This example illustrates that **overloading may also occur in an inheritance relationship**.