Method Overriding

En una jerarquía de clases, cuando un método en una subclase tiene el mismo tipo de retorno y firma que un método en su superclase, se dice que el método en la subclase invalida el método en la superclase. Cuando se llama a un método anulado desde una subclase, siempre se referirá a la versión de ese método definida por la subclase. La versión del método definido por la superclase se ocultará.

```
class A {
   int i, j;
   A(int a, int b) {
      i = a;
      j = b;
   }
   void show() {
   }
}
class B extends A {
   int k;
   B(int a, int b, int c) {
      super(a, b);
      k = c;
   }
   void show() {
      System.out.println("k: " + k);
   }
}
class B2 extends A {
   int k;
   B2(int a, int b, int c) {
      super(a, b);
      k = c;
   }
   void show() {
      super.show(); // this calls A's show()
      System.out.println("k: " + k);
   }
}
class Override {
   public static void main(String args[]) {
      B subOb = new B(1, 2, 3);
      subOb.show(); // this calls show() in B
```

```
    }
}
```

Los métodos con firmas diferentes se sobrecargan y no se anulan

```
class A {
   int i, j;
   A(int a, int b) {
      i = a;
      j = b;
   }
   void show() {
      System.out.println("i and j: " + i + " " + j);
   }
}
class B extends A {
   int k;
   B(int a, int b, int c) {
      super(a, b);
      k = c;
   }
   void show(String msg) {
      System.out.println(msg + k);
   }
}
class Overload {
   public static void main(String args[]) {
      B subOb = new B(1, 2, 3);
      subOb.show("This is k: "); // this calls show() in B
      subOb.show(); // this calls show() in A
   }
}
```

El poder del polimorfismo

La anulación de métodos constituye la base de uno de los conceptos más poderosos de Java: el envío dinámico de métodos. El envío dinámico de métodos es el mecanismo mediante el cual una llamada a un método anulado se resuelve en tiempo de ejecución en lugar de compilar. El envío dinámico de métodos es importante porque así es como Java implementa el polimorfismo en tiempo de ejecución.

```
class Sup {
   void who() {
      System.out.println("who() in Sup");
   }
}
class Sub1 extends Sup {
   void who() {
      System.out.println("who() in Sub1");
   }
}
class Sub2 extends Sup {
   void who() {
      System.out.println("who() in Sub2");
   }
}
class DynDispDemo {
   public static void main(String args[]) {
      Sup superOb = new Sup();
      Sub1 subOb1 = new Sub1();
      Sub2 subOb2 = new Sub2();
      Sup supRef;
      supRef = superOb;
      supRef.who(); //Sup
      supRef = subOb1;
      supRef.who();//Sub1
      supRef = subOb2;
      supRef.who();//Sub2
   }
}
```