Supplement: Extended Discussion on Overriding vs. Overloading
For Introduction to Java Programming
By Y. Daniel Liang

You have learned about overloading methods in Chapter 5. Overloading a method is a way to provide more than one method with the same name but with different signatures to distinguish them. To override a method, the method must be defined in the subclass using the same signature and same return type as in its superclass.

Let us use an example to show the differences between overriding and overloading. In (a), the method **p(int i)** in class **A** overrides the same method defined in class **B**. However, in (b), the method **p(double i)** in class **A** and the method **p(int i)** in class **B** are two overloaded methods. The method **p(int i)** in class **B** is inherited in **A**.

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
  }
}

class B {
  public void p(int i) {
  }
}

class A extends B {
  // This method overrides the method in B
  public void p(int i) {
    System.out.println(i);
  }
}
```

(a)

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
  }
}

class B {
  public void p(int i) {
  }
}

class A extends B {
  // This method overloads the method in B
  public void p(double i) {
    System.out.println(i);
  }
}
```

(b)

When you run the **Test** class in (a), **a.p(10)** invokes the **p(int i)** method defined in class **A**, so the program displays **10**. When you run the **Test** class in (b), **a.p(10)** invokes the **p(int i)** method defined in class **B**, so nothing is printed.

> **NOTE**
> You can override a method with a compatible return type. The new type must be the same type or a subtype of the original type. For example, the following code in (a) is correct, but it is wrong in (b).

```java
public class Test {
  public static void main(String[] args) {
    A a = new A();
    System.out.println(a.getValue());
  }
}

class B {
  public Object getValue() {
    return "Any object";
  }
}

class A extends B {
  public String getValue() {
    return "A string";
  }
}
```

(a) Compatible type

```java
public class Test {
  public static void main(String[] args) {
    A a = new A();
    System.out.println(a.getValue());
  }
}

class B {
  public String getValue() {
    return "Any object";
  }
}

class A extends B {
  public Object getValue() {
    return "A string";
  }
}
```

(b) Incompatible type