说说 Java 8 新特性，default 方法

作者：zjhch123
原文链接：https://ld246.com/article/1481123632905
来源网站：链滴

Java 8 新特性，default方法

default方法，也可称为Defender方法，或者虚拟拓展方法(Virtual extension methods)。

1. 来源

Java8的一个重要的特性就是引入了函数式方法，其中Collection接口中增加了新的stream()方法。

我们都知道在Java的接口中只能定义方法，而不能对方法进行具体实现。其方法的实现必须要到实现该接口的非抽象子类中实现。

因为接口导致的这个语法限制，使得要实现Collection接口的stream()方法全覆盖变得异常困难。难所有继承了Collection接口的类中都要重写stream()方法的实现吗？显然不是的。机智的Java开发工师们在Java8中引入了这样一个新的概念——'default'方法。

简单的说，Java接口拥有了非抽象方法了！default方法带来的好处是可以往接口中新增加方法，而破坏现有的实现架构。

2. 使用

2.1 HelloWorld

从最简单的例子说起

```
interface Father {
    default public void sayHi() {
        System.out.println("Hi Father");
    }
}

class FatherImpl implements Father {

}

public class Default {
    public static void main(String[] args) {
        Father f = new FatherImpl();
        f.sayHi();
    }
}
```

以上代码，声明了一个Father接口，接口中包含一个default方法sayHi()，其作用是输出一句话。定了一个FatherImpl类实现了Father接口。并在主类中新建了FatherImpl类，调用其sayHi()方法。

其输出结果为：

```
Hi Father
```

2.2 多个default方法

接口中可以定义多个default方法。

```
interface Father {
```

```
    default public void sayHi() {
        System.out.println("Hi Father");
    }

    default public void sayHello() {
        System.out.println("Hello Father");
    }
}

class FatherImpl implements Father {

}

public class Default {
    public static void main(String[] args) {
        Father f = new FatherImpl();
        f.sayHi(); // Hi Father
        f.sayHello(); // Hello Father
    }
}
```

2.3 重写default方法

default方法一旦与类中定义的方法有冲突，编译器优先选择类中定义的方法。如下：

```
interface Father {
    default public void sayHi() {
        System.out.println("Hi Father");
    }
}

class FatherImpl implements Father {
    public void sayHi() {
        System.out.println("Hi FatherImpl");
    }
}

public class Default {
    public static void main(String[] args) {
        Father f = new FatherImpl();
        f.sayHi(); // Hi FatherImpl
    }
}
```

2.4 同时继承和实现

以下是一种特殊情况：

```
interface Father {
    default public void sayHi() {
        System.out.println("Hi Father");
    }
}
```

```
class FatherImpl implements Father {
    public void sayHi() {
        System.out.println("Hi FatherImpl");
    }
}

class SubFatherImpl extends FatherImpl implements Father {

}

public class Default {
    public static void main(String[] args) {
        SubFatherImpl f = new SubFatherImpl();
        f.sayHi(); // Hi FatherImpl
    }
}
```

SubFatherImpl中没有做任何其他操作，只是继承了FatherImpl，实现了Father接口。最后通过f.sayH()调用到的是FatherImpl中的方法，而不是接口中定义的default方法，原因在于，与接口中定义的默方法相比，类中定义的方法更具体。

2.4 能说明，类中重写的方法优先级大于接口中的默认方法。这样的设计主要是由增加默认方法的目决定的。

增加默认方法的目的是为了在接口上进行向后兼容。

假设已实现了一个继承于List接口的定制的列表类MyList，该类中有一个addAll()方法，如果新的List口增加了默认方法addAll()。如果类中重写的方法没有默认方法的优先级高，那么就会破坏已有的实。—— Java8 函数式编程

2.5 多重继承

当一个类实现了多个接口，其中有某两个接口定义了相同函数签名的默认方法而又没有进行实现时，会引起问题，如下：

```
interface Father {
    default public void sayHi() {
        System.out.println("Hi Father");
    }
}

interface Father2 {
    default public void sayHi() {
        System.out.println("Hi Father2");
    }
}

class FatherImpl implements Father,Father2 {

}
```

FatherImpl类上有一个错误Duplicate default methods named sayHi with the parameters () and (are inherited from the types Father2 and Father，意思大概是说，编译器不明确应该使用接口中

哪个方法。在类中实现相应的方法就可以解决此问题。

如下：

```
interface Father {
    default public void sayHi() {
        System.out.println("Hi Father");
    }
}

interface Father2 {
    default public void sayHi() {
        System.out.println("Hi Father2");
    }
}

class FatherImpl implements Father,Father2 {
    public void sayHi() {
        System.out.println("No!! It's my method!");
    }
}
```

如果我们想实现指定类的默认方法呢?

```
class FatherImpl implements Father,Father2 {
    public void sayHi() {
        Father2.super.sayHi();
    }
}
```

当然，我们可以同时使用两个接口的default方法

```
interface Father {
    default public void sayHi() {
        System.out.println("Hi Father");
    }
}

interface Father2 {
    default public void sayHi() {
        System.out.println("Hi Father2");
    }
}

class FatherImpl implements Father,Father2 {
    public void sayHi() {
        Father2.super.sayHi();
        Father.super.sayHi();
    }
}

public class Default {
    public static void main(String[] args) {
        FatherImpl f = new FatherImpl();
        f.sayHi();
```

```
        //输出 Hi Father2 \n Hi Father
    }
}
```

3. 三定律

1. 类优先级高于接口。如果在继承链中有方法体或抽象的方法声明，那么就可以忽略接口中定义的方
。

2. 子类优先级高于父类。如果一个接口继承了另一个接口，且两个接口都定义了一个默认方法，那么
类中定义的方法胜出。

3. 如果上面两条规则都不适用，子类要么需要实现该方法，要么将该方法声明为抽象方法。

4. 权衡

在接口中定义方法的诸多变化引发了一系列问题。

既然可以用代码主体定义方法，那Java8种的接口还是旧版本中界定的代码吗?

我觉得，不到万不得已的时候不要使用default。在对问题域建模时，还是应该根据具体情况进行权衡。

我目前所接触到的使用default方法的地方是SpringDataJPA使用QueryDsl时的场景，其需求是对传
参数进行自定义的绑定操作。详情可参考

Spring Data JPA - Reference

5. 参考

1. 《Java8函数式编程》

2. Java8揭秘(三)Default 方法

3. Java 8新特性——default方法（defender方法）介绍