JAVA 达摩班

JAVA 8

季蠡
Ashton

07 April, 2018

接口改进

Lambda表达式

流式API

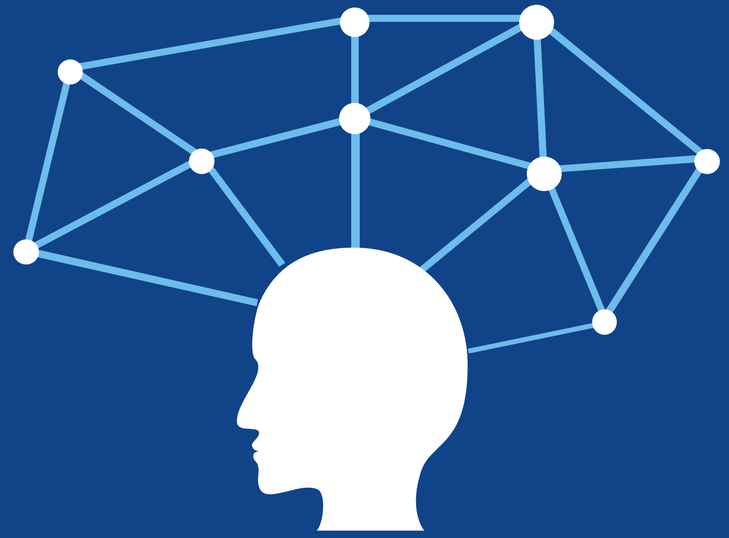
日期时间API改进

迭代器改进

并发API改进

集合操作改进

文件操作改进



http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html

函数式接口与默认方法

1

回顾

抽象类

现实存在不能独立存在的基类
e.g. Animal: Cat, Cow, Dog

1. 包含所有子类的共有属性
2. 数据成员
3. 实现方法
4. 抽象方法，必须被复写
5. 不能实例化

```
abstract class Shape {
    protected int x, y;
    Shape(int _x, int _y) {
        x = _x;
        y = _y;
    }
    abstract public void draw();
    abstract public void erase();
    public void moveTo(int _x, int _y) {
        erase();
        x = _x;
        y = _y;
        draw();
    }
}
```

```
class Circle extends Shape {
    private int r;
    public Circle(int _x, int _y, int _r) {
        super(_x, _y);
        r = _r;
        draw();
    }
    public void draw() {
        System.out.println("Draw circle at ("+x+","+y+")");
    }
    public void erase() {
        System.out.println("Erase circle at ("+x+","+y+")");
    }
}
```

回顾

接口

接口规定一个概念的规约或行为

不允许多重继承，即类**不能有多个父类**
允许实现多个接口

1. 接口不提供实现，定义契约或协议
2. 接口可以看作“纯”抽象类，只有抽象方法
3. 基于“has-a”拥有关系
4. 实现类必须实现所有

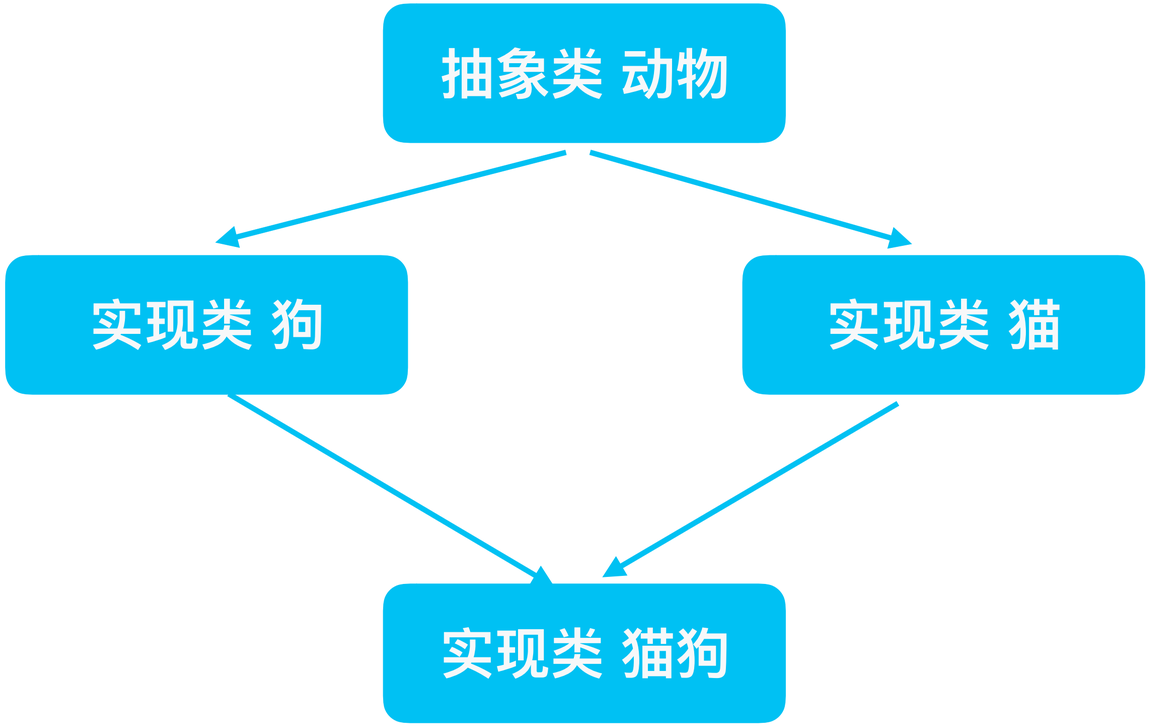
```
interface CanFight {
    void fight();
}
interface CanSwim {
    void swim();
}
interface CanFly {
    void fly();
}
class ActionCharacter {
    public void fight() {...}
}
```

```
class Hero extends ActionCharacter
implements CanFight, CanSwim, CanFly {
    public void swim() {...}
    public void fly() {...}
}
```

回顾

```
public abstract class Animal {
  public abstract void noise();
}

public class Dog extends Animal{
  @Override
  public void noise(){
    System.out.println("woof");
  }

  public void methodA(){}
}

public class Cat extends Animal{
  @Override
  public void noise(){
    System.out.println("meow");
  }

  public void methodB(){}
}
```

菱形问题




```
public class DC extends Dog, Cat{
  public void test(){
    noise();
  }
}
```

回顾

抽象类	接口
不完整的类，可以包含部分实现，必须”特殊化“才能使用	规定概念的形式/契约，而不提供任何实现
可以有数据成员	不能有数据成员，只能有常量
通过继承实现层次化的类结构	轻量级实现
单继承	多重实现
is-a 继承关系	has-a 组合关系
可以有构造函数	没有构造函数
成员可以为public，private或protected	不能使用访问控制符，默认public
实现方法可以为静态	不能有静态定义
抽象类可以继承其他类，并且实现接口	接口可以继承另一个接口

默认方法

Java 8 在接口中使用default定义的方法
防御方法（defender method），或虚拟扩展方法（virtual extention method）

1. 不一定要复写默认方法
2. 如果类实现两个拥有同名默认方法的接口，那么该默认方法必须要复写
3. 默认方法使得接口和抽象类的区别更加微妙
4. 共同的工具方法可以定义为默认方法
5. 避免基础实现类（接口的第一层基础实现）
6. 默认方法不能复写Object类的方法

静态方法

Java 8 在接口中使用static定义的方法
静态方法和默认方法一样，除了静态方法**不能被复写**

1. 用于避免不好的复写实现，也就免掉部分安全问题
2. 接口静态方法仅对接口可见，即不能在接口实现类对象上调用
3. 更适合做工具方法
4. 可以减少工具类的数量，相应的方法可以改为接口静态方法，便于寻找和使用

函数式接口

如果一个接口只有一个抽象方法，那么它叫做函数式接口（Functional Interface）

@FunctionalInterface注释用于标示该接口为函数式接口，避免多余抽象方法

函数式接口允许使用lambda表达式实例化，这是Java 8的一个重要特性

java.util.function e.g.Consumer, Supplier, Function and Predicate

Java 8 重写Collection API，新的Stream API提供大量函数式接口

函数式接口和Lambda表达式

```
@FunctionalInterface
public interface Runnable {
    void run();
}

Runnable r = new Runnable(){
                @Override
                public void run() {
                        System.out.println("My Runnable");
                }
};

Runnable r1 = () -> System.out.println("My Runnable");
```

Lambda 表达式深入

2

Lambda表达式

匿名函数 = (argument) -> (body)

意义：

1. 减少代码行数。相比匿名（内部）类，使用lambda表达式实例化函数式接口更简洁
2. Stream API + Lambda 实现并行/串行操作
3. 传递lambda作为方法参数
4. 惰性计算，或延迟计算（lazy evaluation）

Lambda表达式

1. 不能有函数名，代表匿名类
2. 代表函数式接口实例
3. 返回值类型由编译器“推测”
4. 不能出现泛型，泛型定义在函数式接口中
5. 表达式内变量必须是final，不能在表达式内修改。或者显示定义为final，或者“effectively final“变量，即默认final（local-non-final-initialized-only-once ）
6. 没有自己的作用域，this或super指向lambda表达式外部
7. 表达式内部可以使用break, continue, return和throw等

Lambda表达式

```
() -> {}                        // void
() -> 42                        // 表达式
() -> { return 42; }            // 有返回值，函数体块
() -> { System.gc(); }          // 无返回值，函数体块

(int x) -> x+1                  // 显示声明参数（declared types），表达式等价于返回值
(int x) -> { return x+1; }
(x) -> x+1                      // 隐式声明参数（inferred-type）
x -> x+1                        // 单隐式声明参数，括号可省略

(int x, int y) -> x+y
(x,y) -> x+y
(x, final y) -> x+y             // 错误，隐式参数不能修改
(x, int y) -> x+y               // 错误，混用参数
```

Lambda表达式

方法引用：引用类方法而不执行，类似取方法名
构造引用：引用类构造方法而不实例化类，类似取构造方法名

方法引用类型：

1. TypeName::staticMethod - 类，接口，枚举的静态方法
2. objectRef::instanceMethod - 对象实例方法
3. ClassName::instanceMethod - 类的实例方法
4. TypeName.super::instanceMethod - 超类的实例方法
5. ClassName::new - 类构造方法
6. ArrayTypeName::new - 数组类型的构造方法

System::getProperty
System.out::println
“abc"::length

ArrayList::new
int[]::new

Lambda表达式

Java 8 引入新类型：**交叉类型（intersection type）**

1. 多个类型的交叉类型
2. 类型转换的结果
3. Type = Type1 & Type2

函数接口（Function<T, R>）：代表一个函数，接受一个参数返回一个结果
四个方法：apply，compose，andThen和identify
包括六个实现：IntFunction<R>, LongFunction<R>, DoubleFunction<R>, ToIntFunction<T>, ToLongFunction<T>, ToDoubleFunction<T>

Iterator 进阶及应用

3

Stream API 进阶及应用

4

Date和Time 进阶及应用

5

Stream API 进阶及应用

6

作业

1. 杂货店程序多线程版

2. 查阅和理解volatile变量，以及在线程操作的应用

3. 总结executorService的execute，submit和schedule方法

Thanks!

Any questions?