Java 核心技术(高阶)

第二章 Java泛型

第四节 泛型实现的本质和约束

华东师范大学 陈良育

泛型

- 泛型：编写的代码可以被很多不同类型的对象所重用
  - 泛型类： ArrayList，HashSet，HashMap等
  - 泛型方法：Collections.binarySearch，Arrays.sort 等
  - 泛型接口： List, Iterator 等
- 泛型是JDK1.5引进的新特性
- JDK的版本是向后兼容的
  - 即低版本的class文件可以在高版本的JDK上运行
- 因此，JVM里面没有泛型对象，而是采用类型擦除技术，只有普通的类和方法

类型擦除(1)

- 擦除泛型变量，替换为原始类型(raw type)，无限定为Object

```
public class Pair<T> {

    private T first;
    private T second;

    public Pair(T first, T second) {
        this.first = first;
        this.second = second;
    }

    public T getFirst() {
        return first;
    }

    public void setFirst(T first) {
        this.first = first;
    }
}
```

```
public class Pair {

    private Object first;
    private Object second;

    public Pair(Object first, Object second) {
        this.first = first;
        this.second = second;
    }

    public Object getFirst() {
        return first;
    }

    public void setFirst(Object first) {
        this.first = first;
    }
}
```

类型擦除(2)

- 擦除泛型变量，替换为原始类型(raw type)，有限定则为第一个类型

```
public class NewPair<T extends Comparable & Serializable> {

    private T first;
    private T second;

    public NewPair(T first, T second) {
        this.first = first;
        this.second = second;
    }

    public T getFirst() {
        return first;
    }

    public void setFirst(T first) {
        this.first = first;
    }
}
```

```
public class NewPair {

    private Comparable first;
    private Comparable second;

    public NewPair(Comparable first, Comparable second) {
        this.first = first;
        this.second = second;
    }

    public Comparable getFirst() {
        return first;
    }

    public void setFirst(Comparable first) {
        this.first = first;
    }
}
```

类型擦除(3)

- 擦除泛型变量后，为了保证类型的安全性，需要自动进行类型转换
- 泛型表达式翻译
  - Fruit a = fruits.getFirst();
  - Object a1 = fruits.getFirst();
  - Fruit a = (Fruit) a1;

类型擦除(4)

- 擦除泛型变量后，为了保证类型的安全性，需要自动进行类型转换
- 泛型方法翻译

```
public void setFirst(T first) {
    this.first = first;
}
```

```
public void setFirst(T first);
    Flags: PUBLIC
    Code:
          linenumber      18
       0: aload_0         /* this */
       1: aload_1         /* first */
       2: putfield        overloadtest/Pair.first:Ljava/lang/Object;
          linenumber      19
       5: return
```

类型擦除(5)

- 重载泛型方法翻译(自动桥方法)

```
public class IntPair extends Pair<Integer> {

    public IntPair(Integer first, Integer second) {
        super(first, second);
    }

    public void setFirst(Integer first) {
        super.setFirst(first);
    }

    public static void main(String[] args) {
        IntPair nums = new IntPair(1, 2);
        Pair<Integer> nums2 = nums; // 转为父类对象
        nums2.setFirst(3); // 多态调用本类的setFirst方法
    }
}
```

```
public void setFirst(java.lang.Integer first);
    Flags: PUBLIC
    Code:
          linenumber      10
        0: aload_0        /* this */
        1: aload_1        /* first */
        2: invokespecial  overloadtest/Pair.setFirst:(Ljava/lang/Object;)V
          linenumber      11
        5: return

public void setFirst(java.lang.Object p0);
    Flags: PUBLIC, BRIDGE, SYNTHETIC
    Code:
          linenumber      1
        0: aload_0
        1: aload_1
        2: checkcast      Ljava/lang/Integer;
        5: invokevirtual  overloadtest/IntPair.setFirst:(Ljava/lang/Integer;)V
        8: return
```

泛型的约束

- 不能用基本类型(8种)来实例化泛型
- 运行时类型查询只适用于原始类型
- 不能创建参数化类型的数组
- 可变参数警告
- 不能实例化类型变量
- 不能构造泛型数组
- 泛型类的静态上下文中类型变量无效
- 不能抛出或捕获泛型类的异常实例
- 可以消除对受查异常(checked exception)的检查
- 类型擦除后引发的方法冲突

总结

- 泛型：类型擦除
  - 虚拟机中没有泛型，只有普通类和方法
  - 在编译阶段，泛型参数被擦除为限定类型，并进行相关类型转换
  - 虚拟机也会合成桥方法来保持方法多态

谢 谢！