Java 核心技术

第六章　static、final和常量设计

第四节 常量设计和常量池

华东师范大学 陈良育

常量设计(1)

- 常量：一种不会修改的变量
  - Java没有constant关键字
  - 不能修改，final
  - 不会修改/只读/只要一份，static
  - 方便访问public
- Java中的常量
  - public static final
  - 建议变量名字全大写，以连字符相连，如UPPER_BOUND

常量设计(2)


```
public class Constants {
    public final static double PI_NUMBER = 3.14;
    public static final String DEFAULT_COUNTRY="China";

    public static void main(String[] a)
    {
        System.out.println(Constants.PI_NUMBER);
        System.out.println(Constants.DEFAULT_COUNTRY);
    }
}
```

常量设计(3)

- 一种特殊的常量：接口内定义的变量默认是常量

```
public interface SpecialAnimal {
     String color = "yellow"; //default: public static final
     public void move();
}
```

```
public class Cat implements SpecialAnimal {
    public void move() {
        System.out.println("I can move");
    }

    public static void main(String[] args) {
        Cat cat = new Cat();
        cat.color = "white";  //error, the variables in interface are constants.
    }
}
```

常量池(1)

- 首先请猜一猜下列程序的输出结果

```
public class IntegerTest {
    public static void main(String[] args) {
        Integer n1 = 127;
        Integer n2 = 127;
        System.out.println(n1 == n2);
        Integer n3 = 128;
        Integer n4 = 128;
        System.out.println(n3 == n4);
        Integer n5 = new Integer(127);
        System.out.println(n1 == n5);
    }
}
```

常量池(2)

- Java为很多基本类型的包装类/字符串都建立常量池
- 常量池：相同的值只存储一份，节省内存，共享访问
- 基本类型的包装类
  - Boolean，Byte，Short，Integer，Long，Character，Float，Double
  - Boolean： true, false
  - Byte, Character : \u0000--\u007f (0—127)
  - Short, Int, Long： -128~127
  - Float，Double：没有缓存(常量池)
  - 查看CacheTest.java

常量池(3)

- Java为常量字符串都建立常量池缓存机制
- 字符串常量

```
public class StringConstantTest {
    public static void main(String[] args) {
        String s1 = "abc";
        String s2 = "abc";
        String s3 = "ab" + "c"; //都是常量，编译器将优化，下同
        String s4 = "a" + "b" + "c";
        System.out.println(s1 == s2); //true
        System.out.println(s1 == s3); //true
        System.out.println(s1 == s4); //true
    }
}
```

常量池(4)

- 基本类型的包装类和字符串有两种创建方式
  - 常量式(字面量)赋值创建，放在栈内存 (将被常量化)
    - Integer a = 10;
    - String b = “abc”;
  - new对象进行创建，放在堆内存 (不会常量化)
    - Integer c = new Integer(10);
    - String d = new String(“abc”);
- 这两种创建方式导致创建的对象存放的位置不同

常量池(5)

- 查看BoxClassTest.java 分析Integer类
  - 基本类型和包装类比较，将对包装类自动拆箱
  - 对象比较，比较地址
  - 加法+会自动拆箱
- 查看StringNewTest.java 分析String类
  - 常量赋值(堆内存)和new创建(栈内存)不是同一个对象
  - 编译器只会优化确定的字符串，并缓存

总结

- Java中的常量：static和final
- Java接口中的变量都是常量
- 对象生成有两种：常量赋值(栈内存)和new创建(堆内存)
- Java为Boolean, Byte, Character, Short, Int, Long, String 的常量赋值建立常量池，没有包括Float和Double
- Java编译器会优化已经确定的变量

代码(1) Constants.java


```
public class Constants {
    public final static double PI_NUMBER = 3.14;
    public static final String DEFAULT_COUNTRY="China";

    public static void main(String[] a)
    {
        System.out.println(Constants.PI_NUMBER);
        System.out.println(Constants.DEFAULT_COUNTRY);
    }
}
```

代码(2) SpecialAnimal&Cat.java


```
public interface SpecialAnimal {
     String color = "yellow"; //default: public static final
     public void move();
}
```

```
public class Cat implements SpecialAnimal {
    public void move() {
        System.out.println("I can move");
    }

    public static void main(String[] args) {
        Cat cat = new Cat();
        cat.color = "white";  //error, the variables in interface are constants.
    }
}
```

代码(3) IntegerTest.java


```java
public class IntegerTest {

    public static void main(String[] args) {
        Integer n1 = 127;
        Integer n2 = 127;
        System.out.println(n1==n2);
        //对象双等号是比较指针是否指向同一个东西

        Integer n3 = 128;
        Integer n4 = 128;
        System.out.println(n3==n4);

        Integer n5 = new Integer(127);
        System.out.println(n1==n5);
    }

}
```

代码(4) CacheTest.java


```java
public class CacheTest {
    public static void main(String[] args) {
        Boolean b1 = true;  //true,false
        Boolean b2 = true;
        System.out.println("Boolean Test: " + String.valueOf(b1 == b2));

        Byte b3 = 127;      //\u0000-\u007f
        Byte b4 = 127;
        System.out.println("Byte Test: " + String.valueOf(b3 == b4));

        Character c1 = 127;  //\u0000-\u007f
        Character c2 = 127;
        System.out.println("Character Test: " + String.valueOf(c1 == c2));

        Short s1 = -128;  //-128~127
        Short s2 = -128;
        System.out.println("Short Test: " + String.valueOf(s1 == s2));

        Integer i1 = -128;  //-128~127
        Integer i2 = -128;
        System.out.println("Integer Test: " + String.valueOf(i1 == i2));

        Long l1 = -128L;  //-128~127
        Long l2 = -128L;
        System.out.println("Long Test: " + String.valueOf(l1 == l2));

        Float f1 = 0.5f;
        Float f2 = 0.5f;
        System.out.println("Float Test: " + String.valueOf(f1 == f2));

        Double d1 = 0.5;
        Double d2 = 0.5;
        System.out.println("Double Test: " + String.valueOf(d1 == d2));
    }
}
```

代码(5) A&B.java


```java
public class A {
    public Integer num = 100;
    public Integer num2 = 128;
    public Character c = 100;
}
```

```java
public class B {
    public Integer num = 100;
    public Integer num2 = 128;
    public Character c = 100;

    public static void main(String[] args) {
        A a = new A();
        B b = new B();
        //Integer -128~127有常量池 true
        System.out.println(a.num == b.num);
        //Integer 128不在常量池     false
        System.out.println(a.num2 == b.num2);
        //Character 0-127在常量池  true
        System.out.println(a.c == b.c);
    }

}
```

代码(6) StringConstantTest.java


```java
public class StringConstantTest {
    public static void main(String[] args) {
        String s1 = "abc";
        String s2 = "abc";
        String s3 = "ab" + "c"; //都是常量，编译器将优化，下同
        String s4 = "a" + "b" + "c";
        System.out.println(s1 == s2); //true
        System.out.println(s1 == s3); //true
        System.out.println(s1 == s4); //true
    }
}
```

代码(7) BoxClassTest.java


```java
public class BoxClassTest {
    public static void main(String[] args)
    {
        int i1 = 10;
        Integer i2 = 10;                    // 自动装箱
        System.out.println(i1 == i2);    //true
        // 自动拆箱  基本类型和包装类进行比较，包装类自动拆箱

        Integer i3 = new Integer(10);
        System.out.println(i1 == i3);  //true
        // 自动拆箱  基本类型和包装类进行比较，包装类自动拆箱

        System.out.println(i2 == i3); //false
        // 两个对象比较，比较其地址。
        // i2是常量，放在栈内存常量池中，i3是new出对象，放在堆内存中

        Integer i4 = new Integer(5);
        Integer i5 = new Integer(5);
        System.out.println(i1 == (i4+i5));    //true
        System.out.println(i2 == (i4+i5));    //true
        System.out.println(i3 == (i4+i5));    //true
        // i4+i5 操作将会使得i4,i5自动拆箱为基本类型并运算得到10.
        // 基础类型10和对象比较，将会使对象自动拆箱，做基本类型比较

        Integer i6 = i4 + i5;  // +操作使得i4,i5自动拆箱，得到10，因此i6 == i2.
        System.out.println(i1 == i6);  //true
        System.out.println(i2 == i6);  //true
        System.out.println(i3 == i6);  //false
    }
}
```

代码(8) StringNewTest.java


```
public class StringNewTest {
    public static void main(String[] args) {
        String s0 = "abcdef";
        String s1 = "abc";
        String s2 = "abc";
        String s3 = new String("abc");
        String s4 = new String("abc");
        System.out.println(s1 == s2); //true 常量池
        System.out.println(s1 == s3); //false 一个栈内存，一个堆内存
        System.out.println(s3 == s4); //false 两个都是堆内存
        System.out.println("============================");

        String s5 = s1 + "def";     //涉及到变量，故编译器不优化
        String s6 = "abc" + "def"; //都是常量 编译器会自动优化成abcdef
        String s7 = "abc" + new String ("def");//涉及到new对象，编译器不优化
        System.out.println(s5 == s6); //false
        System.out.println(s5 == s7); //false
        System.out.println(s6 == s7); //false
        System.out.println(s0 == s6); //true
        System.out.println("============================");


        String s8 = s3 + "def";//涉及到new对象，编译器不优化
        String s9 = s4 + "def";//涉及到new对象，编译器不优化
        String s10 = s3 + new String("def");//涉及到new对象，编译器不优化
        System.out.println(s8 == s9); //false
        System.out.println(s8 == s10); //false
        System.out.println(s9 == s10); //false
    }
}
```

谢 谢！