

# new与clone

C-2 创建: 张林伟, 最后修改: 张林伟 2018-09-21 15:05

new:

分配内存 -> 调用构造函数初始化 -> 返回引用地址

clone:

分配内存 -> 使用原对象进行各个域填充 -> 返回引用地址

性能比较:

SimpleObject

^ 代码块Java

```
1 public class SimpleObject implements Cloneable {
2
3     private int id;
4     private String name;
5
6     public SimpleObject(int id, String name) {
7         this.id = id;
8         this.name = name;
9     }
10
11     @Override
12     public SimpleObject clone() {
13         Object clone = null;
14         try {
15             clone = super.clone();
16         } catch (CloneNotSupportedException e) {
17             e.printStackTrace();
18         }
19         return (SimpleObject) clone;
20     }
21 }
```

ComplexObject

^ 代码块Java

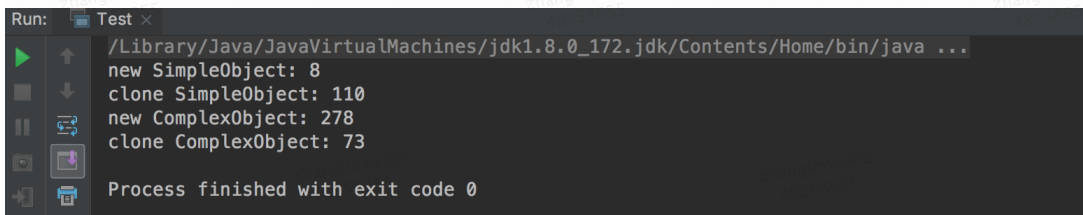
```
1 public class ComplexObject implements Cloneable {
2     private int id;
3     private String name;
4
5     public ComplexObject(int id, String name) {
6         this.id = id;
7         this.name = id + name; // 字符串拼接
8     }
9
10    @Override
11    public ComplexObject clone() {
12        Object clone = null;
13        try {
14            clone = super.clone();
15        } catch (CloneNotSupportedException e) {
16            e.printStackTrace();
17        }
18        return (ComplexObject) clone;
19    }
20 }
```

测试用例

^ 代码块Java

```
1 public class Test {
2
3     public static int COUNT = 10000000;
4
5     public static void main(String[] args) {
6
7         long start = System.currentTimeMillis();
8         for(int i = 0; i < COUNT; i++) {
9             SimpleObject temp = new SimpleObject(1, "simpleObject");
10        }
11        System.out.println("new SimpleObject: " + (System.currentTimeMillis() - start));
12
13        SimpleObject simpleObject = new SimpleObject(1, "simpleObject");
14        start = System.currentTimeMillis();
15        for(int i = 0; i < COUNT; i++) {
16            SimpleObject temp = simpleObject.clone();
17        }
18        System.out.println("clone SimpleObject: " + (System.currentTimeMillis() - start));
19
20        start = System.currentTimeMillis();
21        for(int i = 0; i < COUNT; i++) {
22            ComplexObject temp = new ComplexObject(1, "complexObject");
23        }
24        System.out.println("new ComplexObject: " + (System.currentTimeMillis() - start));
25
26        ComplexObject complexObject = new ComplexObject(1, "complexObject");
27        start = System.currentTimeMillis();
28        for(int i = 0; i < COUNT; i++) {
29            ComplexObject temp = complexObject.clone();
30        }
31        System.out.println("clone ComplexObject: " + (System.currentTimeMillis() - start));
32    }
33 }
```

结果:



```
Run: Test x
/Library/Java/JavaVirtualMachines/jdk1.8.0_172.jdk/Contents/Home/bin/java ...
new SimpleObject: 8
clone SimpleObject: 110
new ComplexObject: 278
clone ComplexObject: 73
Process finished with exit code 0
```

由于对 new 进行了优化,所以在创建简单对象时效率高于 clone;

但对于创建稍复杂的对象, clone 效率要高于 new。

参考资料:

<https://blog.csdn.net/cldance/article/details/77854012>

<https://blog.csdn.net/iblade/article/details/80749148>

仅供内部使用, 未经授权, 切勿外传