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

```java
public class SimpleObject implements Cloneable {

    private int id;
    private String name;

    public SimpleObject(int id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public SimpleObject clone() {
        Object clone = null;
        try {
            clone = super.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
        return (SimpleObject) clone;
    }
}
```

ComplexObject

代码块 Java

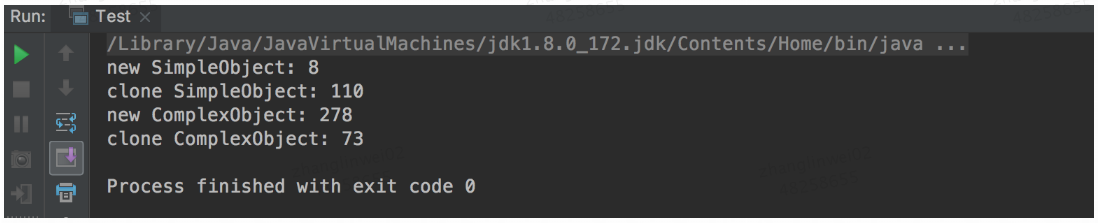
```java
public class ComplexObject implements Cloneable {
    private int id;
    private String name;

    public ComplexObject(int id, String name) {
        this.id = id;
        this.name = id + name;  // 字符串拼接
    }

    @Override
    public ComplexObject clone() {
        Object clone = null;
        try {
            clone = super.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
        return (ComplexObject) clone;
    }
}
```

测试用例

代码块 Java

```java
public class Test {

    public static int COUNT = 10000000;

    public static void main(String[] args) {

        long start = System.currentTimeMillis();
        for(int i = 0; i < COUNT; i++) {
            SimpleObject temp = new SimpleObject(1, "simpleObject");
        }
        System.out.println("new SimpleObject: " + (System.currentTimeMillis() - start));

        SimpleObject simpleObject = new SimpleObject(1, "simpleObject");
        start = System.currentTimeMillis();
        for(int i = 0; i < COUNT; i++) {
            SimpleObject temp = simpleObject.clone();
        }
        System.out.println("clone SimpleObject: " + (System.currentTimeMillis() - start));

        start = System.currentTimeMillis();
        for(int i = 0; i < COUNT; i++) {
            ComplexObject temp = new ComplexObject(1, "complexObject");
        }
        System.out.println("new ComplexObject: " + (System.currentTimeMillis() - start));

        ComplexObject complexObject = new ComplexObject(1, "complexObject");
        start = System.currentTimeMillis();
        for(int i = 0; i < COUNT; i++) {
            ComplexObject temp = complexObject.clone();
        }
        System.out.println("clone ComplexObject: " + (System.currentTimeMillis() - start));
    }
}
```

结果：



```
/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