链滴

java 创建对象的几种方式

作者：bingofang
原文链接：https://ld246.com/article/1516179944842
来源网站：链滴

Java中有5种创建对象的方式，下面给出它们的例子

1、使用new关键字 } → 调用了构造函数

2、使用Class类的newInstance方法 } → 调用了构造函数

3、使用Constructor类的newInstance方法 } → 调用了构造函数

4、使用clone方法 } → 没有调用构造函数

5、使用反序列化 } → 没有调用构造函数

1、最常见的方式使用new关键字:

```
Employee employee = new Employee();
```

2、使用Class类的newInstance方法创建对象。这个newInstance方法调用无参的构造函数创建对象。

```
Employee employee2 = Employee.class.newInstance();
等价于
Employee employee2 = (Employee) Class.forName("com.bingo.eight.Employee").newInstance();
```

3、和Class类的newInstance方法很像， java.lang.reflect.Constructor类里也有一个newInstance方法可以创建对象。我们可以通过这个newInstance方法调用有参数的和私有的构造函数

```
Constructor<Employee> constructor = Employee.class.getConstructor();
Employee employee3 = constructor.newInstance();
```

4、使用clone，无论何时我们调用一个对象的clone方法，jvm就会创建一个新的对象，将前面对象内容全部拷贝进去。用clone方法创建对象并不会调用任何构造函数。要使用clone方法，我们需要实现Cloneable接口并实现其定义的clone方法。

```
Employee employee4 = (Employee) employee3.clone();
```

5.使用反序列化。当我们序列化和反序列化一个对象，jvm会给我们创建一个单独的对象。在反序列时，jvm创建对象并不会调用任何构造函数。为了反序列化一个对象，我们需要让我们的类实现Serializable接口

```
  ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream(new File("obj.txt")));
   Employee employee5 = (Employee) objectInputStream.readObject();
```

下面我们来看一下这个Employee类:

```
 public class Employee implements Cloneable,Serializable{

 private String name;
```

```
    private String age;

    private Integer salary;

    private Status status;

    public Employee() {
        super();
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public Integer getSalary() {
        return salary;
    }

    public void setSalary(Integer salary) {
        this.salary = salary;
    }

    public Status getStatus() {
        return status;
    }
    public void setStatus(Status status) {
        this.status = status;
    }
    public Employee(String name, String age, int salary) {
        super();
        this.name = name;
        this.age = age;
        this.salary = salary;
    }

    public Employee(String name, String age, Integer salary,Status status) {
        this.name = name;
        this.age = age;
        this.salary = salary;
        this.status = status;
    }
    @Override
    public int hashCode() {
```

```
        final int prime = 31;
        int result = 1;
        result = prime * result + ((age == null) ? 0 : age.hashCode());
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        result = prime * result + ((salary == null) ? 0 : salary.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Employee other = (Employee) obj;
        if (age == null) {
            if (other.age != null)
                return false;
        } else if (!age.equals(other.age))
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        if (salary == null) {
            if (other.salary != null)
                return false;
        } else if (!salary.equals(other.salary))
            return false;
        return true;
    }
     @Override
        public Object clone() {
            Object obj = null;
            try {
                obj = super.clone();
            } catch (CloneNotSupportedException e) {
                e.printStackTrace();
            }
            return obj;
        }
    @Override
    public String toString() {
        return "Employee [name=" + name + ", age=" + age + ", salary=" + salary + ", status=" +
tatus + "]";
    }

}
```