实例进行分析。

创建类

分别是Student、Computer、Test

```
public class Student {

        int score;
        int age;
        String name;

        Computer computer;

        public void study() {

                System.out.println("studying...");
        }
}

public class Computer {
        int price;
        String brand;
}

public class Test {

        public static void main(String[] args) {

                Student stu = new Student();

                stu.name = "xiaoming";

                stu.age = 10;

                stu.study();

                Computer c = new Computer();
                c.brand = "Hasse";

                System.out.println(c.brand);

                stu.computer = c;
                System.out.println(stu.computer.brand);
    }
}
```

代码分析

程序的入口是main()，因而从main方法从上到下、从左到右进行分析。

```
Student stu = new Student();
```

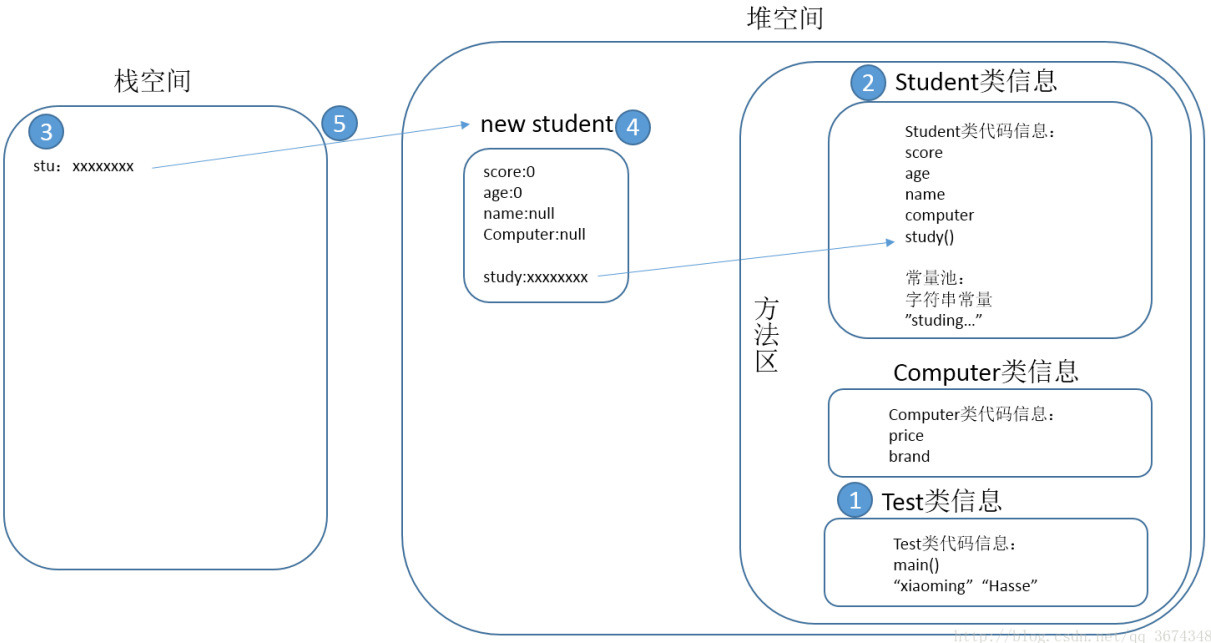
①首先，Java虚拟机（JVM）去方法区寻找是否有Test类的代码信息，如果存在，直接调用。如果没有，通过类加载器（ClassLoader）把.class字节码加载到内存中，并把静态变量和方法、常量池加载（“xiaoming”、“Hasse”）。

②走到Student，以同样的逻辑对Student类进行加载；静态成员；常量池（“studying”）。

③走到stu，stu在main方法内部，因而是局部变量，存放在栈空间中。

④走到new Student，new出的对象（实例），存放在堆空间中，以方法区的类信息为模板创建实例。

⑤‘’ = ‘’ 赋值操作，把new Student的地址告诉stu变量，stu通过四字节的地址（十六进制），引用该实例。




```
stu.name = “xiaoming”;
```

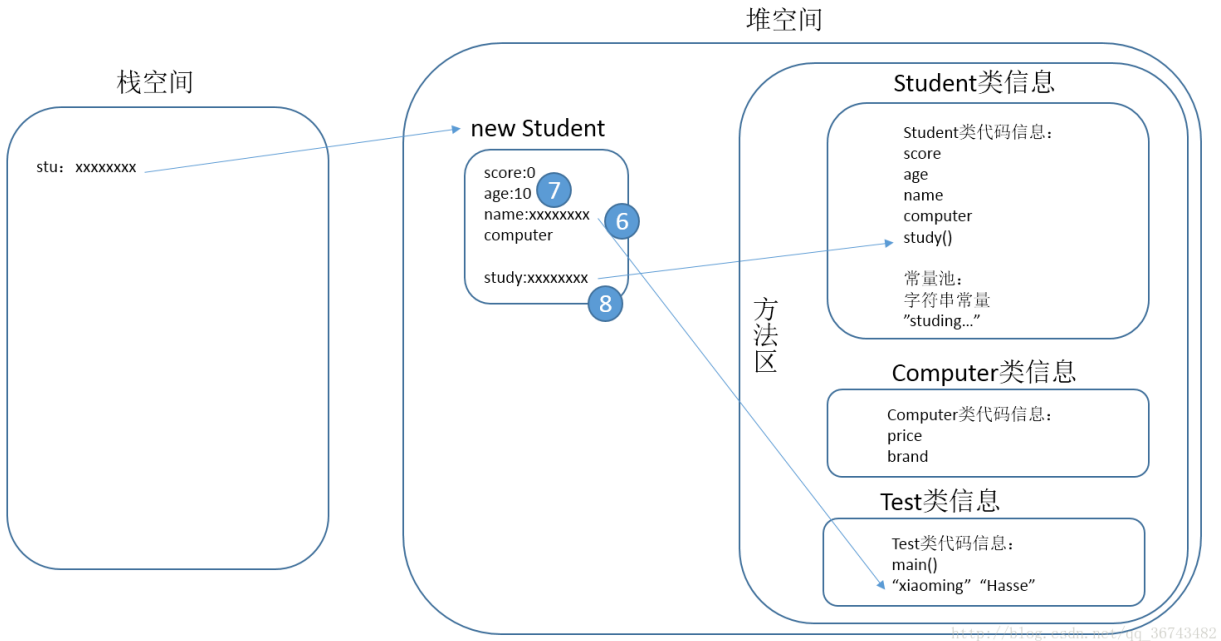
⑥stu通过引用new Student实例的name属性，该name属性通过地址指向常量池的"xiaoming"敞亮。

```
stu.age = 10;
```

⑦s实例的age属性是基本数据类型，基本数据类型直接赋值。

```
stu.study();
```

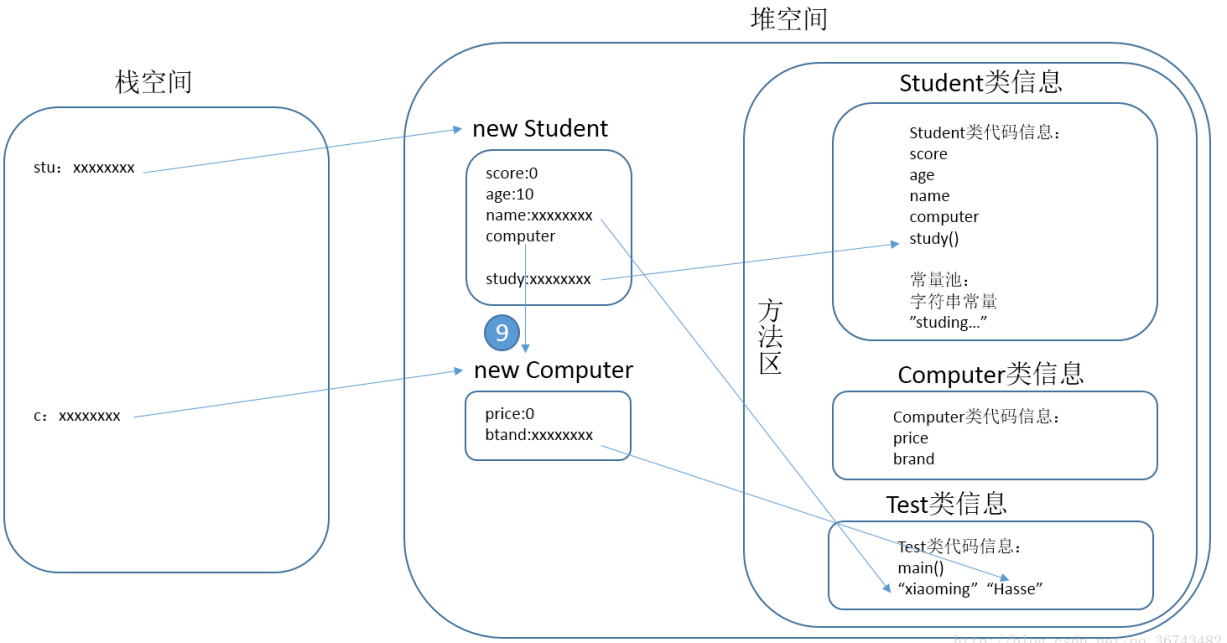
⑧调用实例的方法时，并不会在实例对象中生成一个新的方法，而是通过地址指向方法区中类信息的方法。



Computer c = new Computer();
同stu变量的生成过程。

c.brand = “Hasse”;
同stu.name = ″xiaoming″过程。

stu.computer = c;
⑨把c对象对Computer实例的引用赋值给Student实例的computer属性。
亦即：该Student实例的computer属性指向该Computer类的实例。



拓展

改变brand的地址指向。

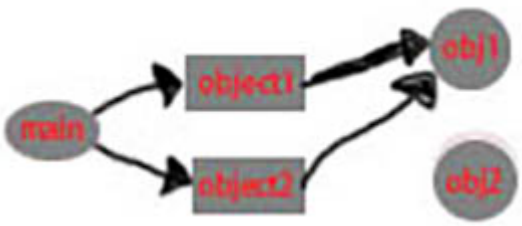
⑨重新将Computer实例的brand属性指向"Dell"常量，那stu.computer.brand指向谁呢？Dell还是Hasse？

```
    c.brand = "Dell";
```

根据刚才的分析可知：
stu通过地址引用Student实例，而该实例的computer的指向和c的指向是同一个Computer实例，因而改变该Computer实例的brand属性的指向，两者都会改变。

两个实例对象。

```
public class Simple {
    public static void main(String args[]){
    Object object1 = new Object();//obj1
    Object object2 = new Object();//obj2
    object2 = object1;
    //...此时，obj2是可以被清理的
    }
}
```



在有向图中，我们叫作obj1是可达的，obj2就是不可达的，显然不可达的可以被清理。

理解字符串常量及常量池

```
                String str = "Dell";
                System.out.println(c.brand == str);
```

根据常量池具有共享性，可知并不会生成新的常量"Dell"，而是会把str通过地址指向原来的"Dell"，因而结果是true。