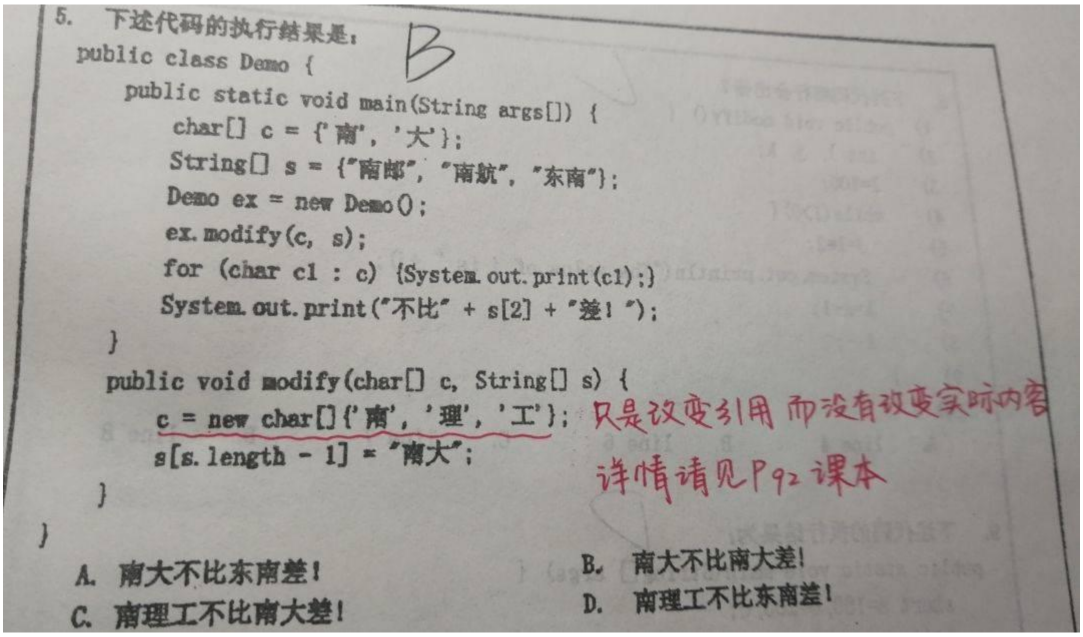


5. 下述代码的执行结果是： B

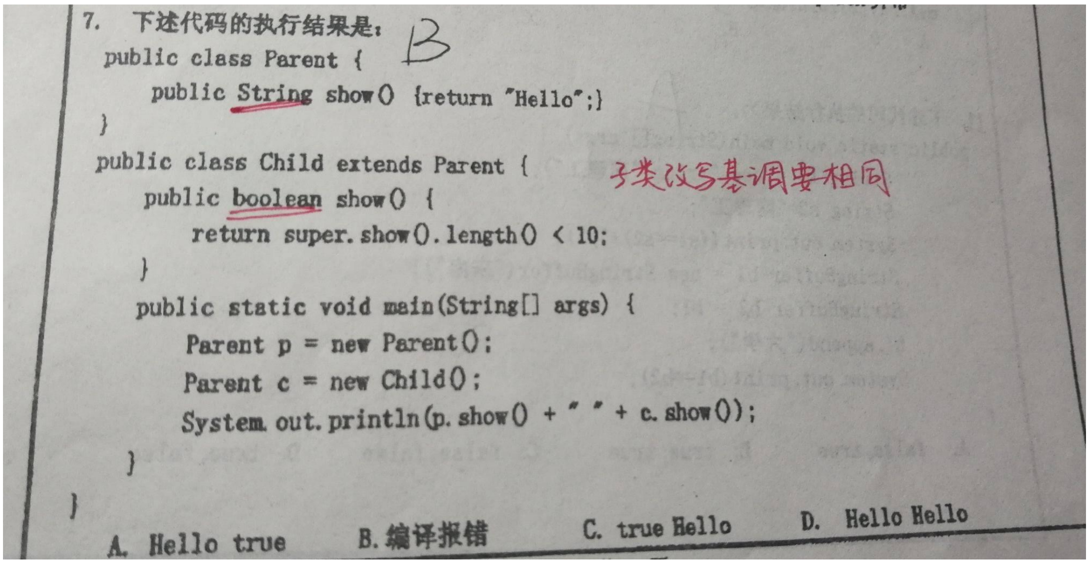
```
public class Demo {
    public static void main(String args[]) {
        char[] c = {'南', '大'};
        String[] s = {"南邮", "南航", "东南"};
        Demo ex = new Demo();
        ex.modify(c, s);
        for (char c1 : c) {System.out.print(c1);}
        System.out.print("不比" + s[2] + "差！");
    }
    public void modify(char[] c, String[] s) {
        c = new char[]{'南', '理', '工'};
        s[s.length - 1] = "南大";
    }
}
```

只是改变引用 而没有改变实际内容

详情请见P92 课本

A. 南大不比东南差！　　B. 南大不比南大差！

C. 南理工不比南大差！　　D. 南理工不比东南差！

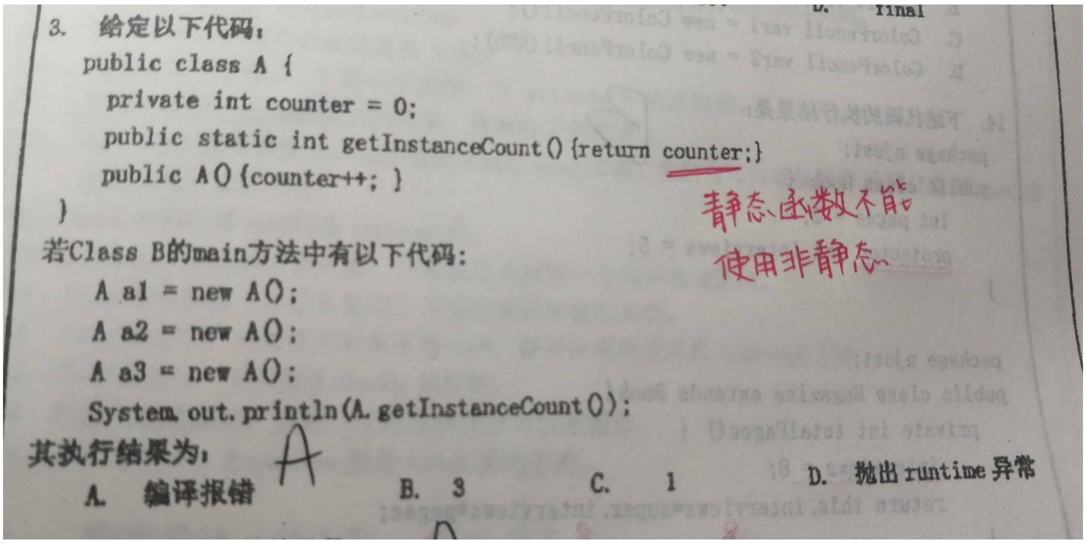


7. 下述代码的执行结果是： B

```
public class Parent {
    public String show() {return "Hello";}
}
public class Child extends Parent {
    public boolean show() {
        return super.show().length() < 10;
    }
    public static void main(String[] args) {
        Parent p = new Parent();
        Parent c = new Child();
        System.out.println(p.show() + " " + c.show());
    }
}
```

子类改写基调要相同

A. Hello true　　B. 编译报错　　C. true Hello　　D. Hello Hello



D.　final

3. 给定以下代码：

```
public class A {
    private int counter = 0;
    public static int getInstanceCount() {return counter;}
    public A() {counter++; }
}
```

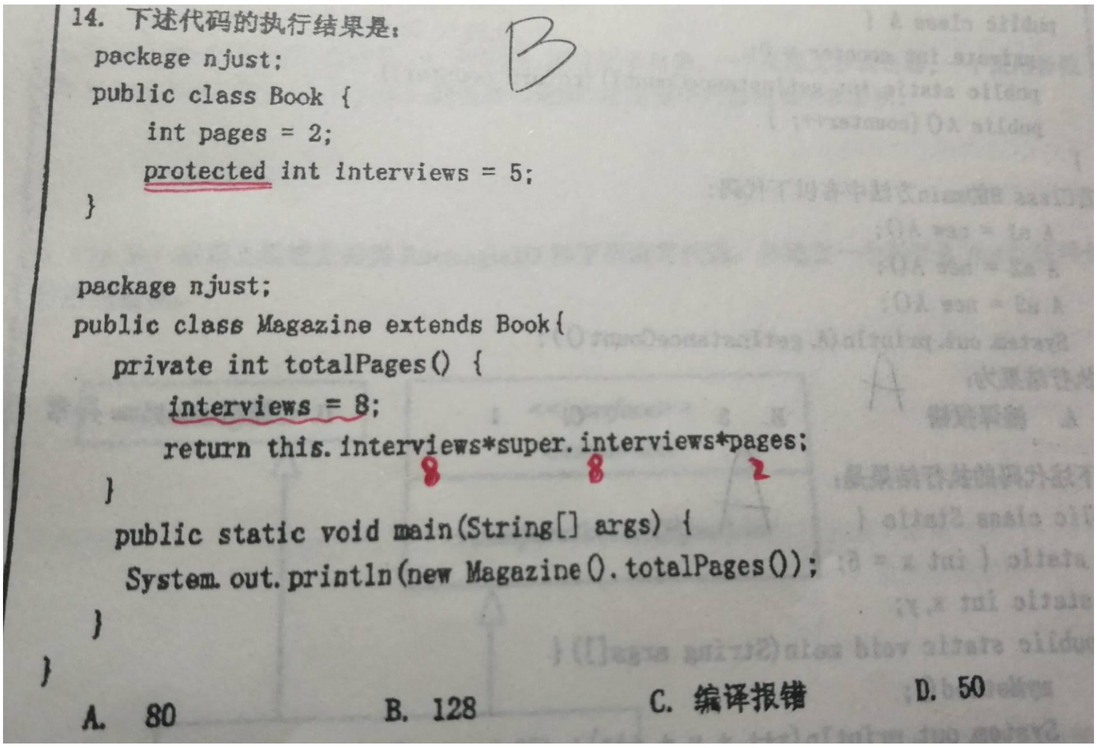
静态函数不能使用非静态

若Class B的main方法中有以下代码：

```
A a1 = new A();
A a2 = new A();
A a3 = new A();
System.out.println(A.getInstanceCount());
```

其执行结果为： A

A. 编译报错　　B. 3　　C. 1　　D. 抛出 runtime 异常

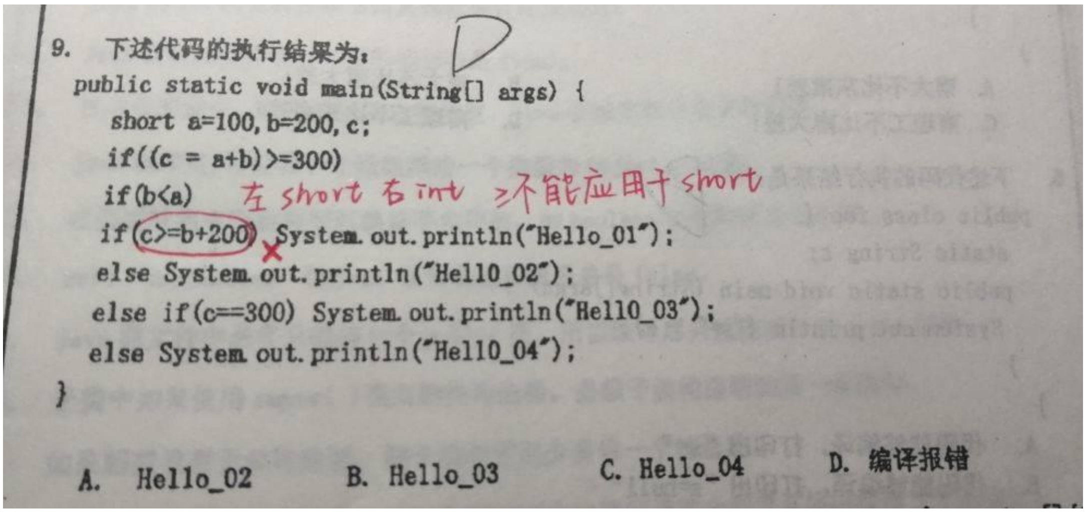


14. 下述代码的执行结果是： B

```
package njust;
public class Book {
    int pages = 2;
    protected int interviews = 5;
}

package njust;
public class Magazine extends Book{
    private int totalPages() {
        interviews = 8;
        return this.interviews*super.interviews*pages;
                        8              8      2
    }
    public static void main(String[] args) {
     System.out.println(new Magazine().totalPages());
    }
}
```

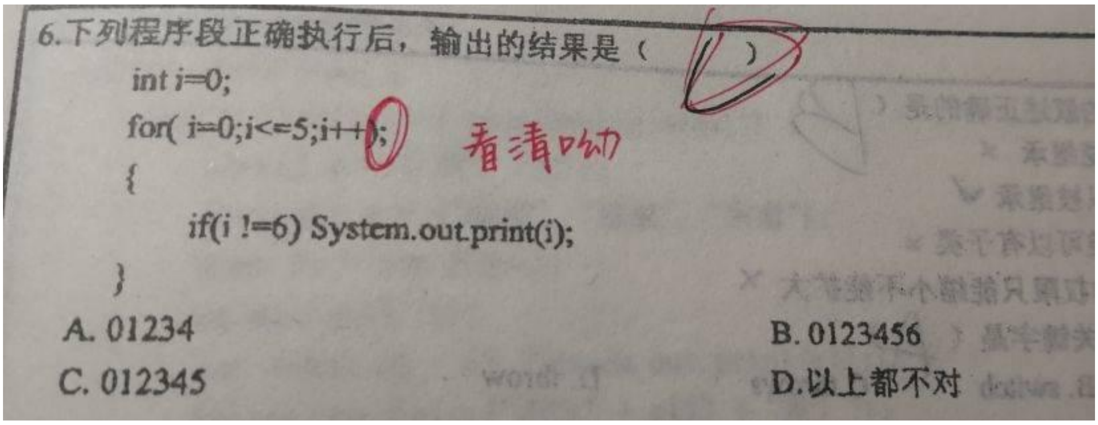
A. 80　　B. 128　　C. 编译报错　　D. 50



9. 下述代码的执行结果为： D

```
public static void main(String[] args) {
    short a=100,b=200,c;
    if((c = a+b)>=300)
    if(b<a)   左short 右int ≥不能应用于short
    if(c>=b+200) ×System.out.println("Hello_01");
   else System.out.println("Hell0_02");
   else if(c==300) System.out.println("Hell0_03");
  else System.out.println("Hell0_04");
}
```

A. Hello_02　　B. Hello_03　　C. Hello_04　　D. 编译报错

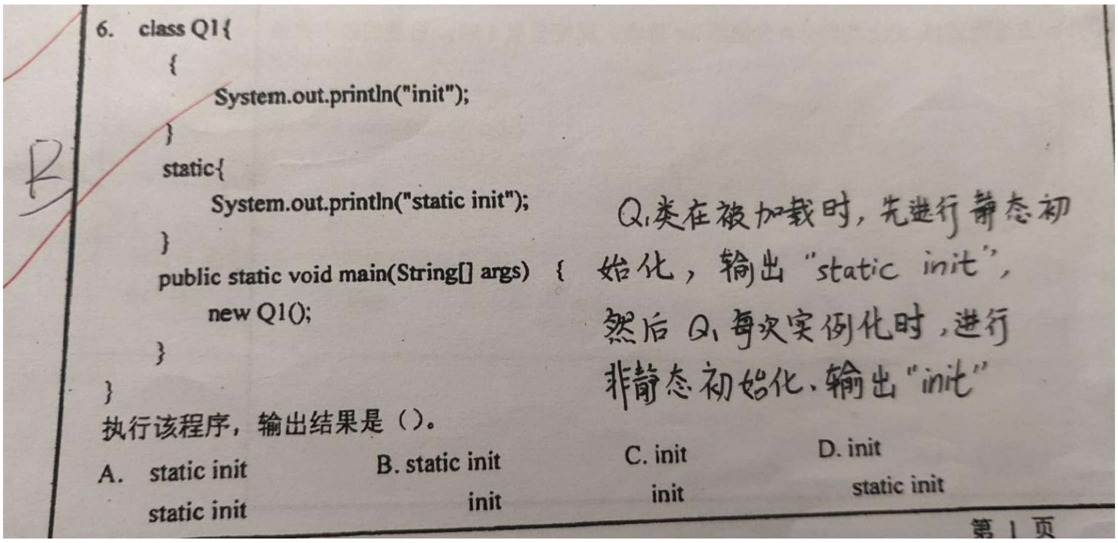


6.下列程序段正确执行后，输出的结果是（ D ）

```
int i=0;
for( i=0;i<=5;i++);   看清呦
{
    if(i !=6) System.out.print(i);
}
```

A. 01234　　B. 0123456

C. 012345　　D.以上都不对



B

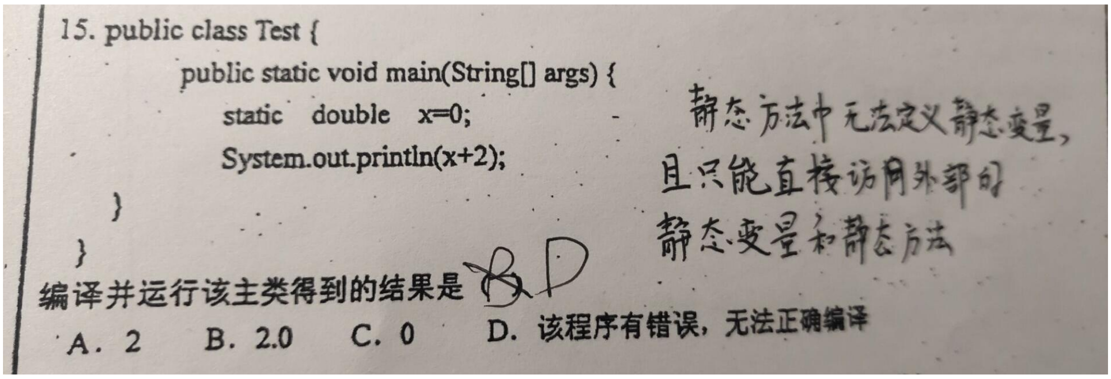
```
6.  class Q1{
        {
            System.out.println("init");
        }
        static{
            System.out.println("static init");
        }
        public static void main(String[] args)  {
            new Q1();
        }
    }
```

执行该程序，输出结果是（）。

A. static init
static init　　B. static init
init　　C. init
init　　D. init
static init

Q₁类在被加载时，先进行静态初始化，输出"static init"，然后Q₁每次实例化时，进行非静态初始化，输出"init"

第 1 页

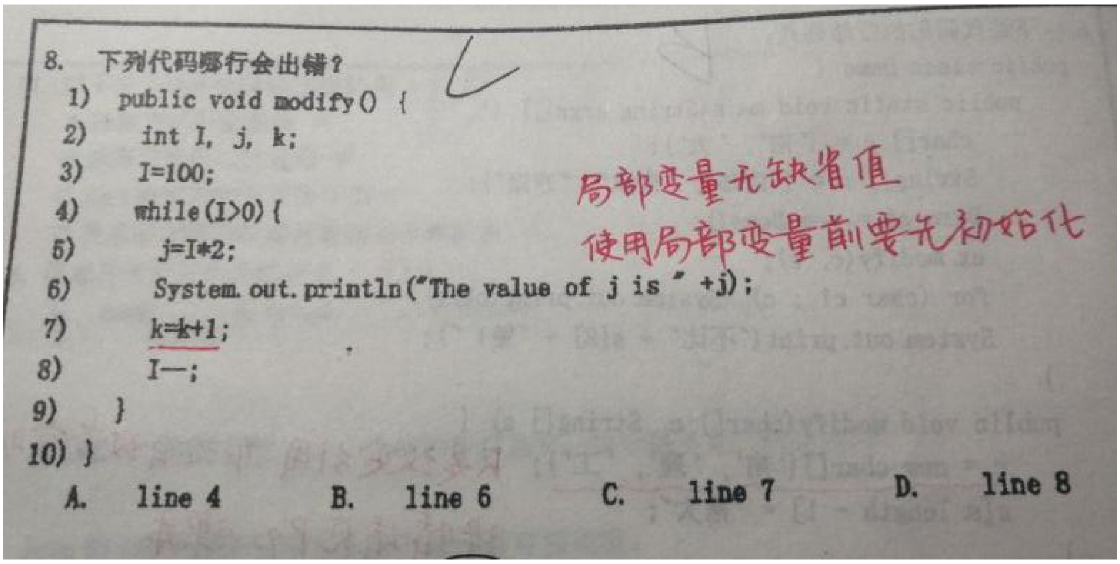


```
15. public class Test {
            public static void main(String[] args) {
                static  double  x=0;
                System.out.println(x+2);
        }
    }
```

编译并运行该主类得到的结果是 ~~B~~D

A. 2　　B. 2.0　　C. 0　　D. 该程序有错误，无法正确编译

静态方法中无法定义静态变量，且只能直接访问外部的静态变量和静态方法



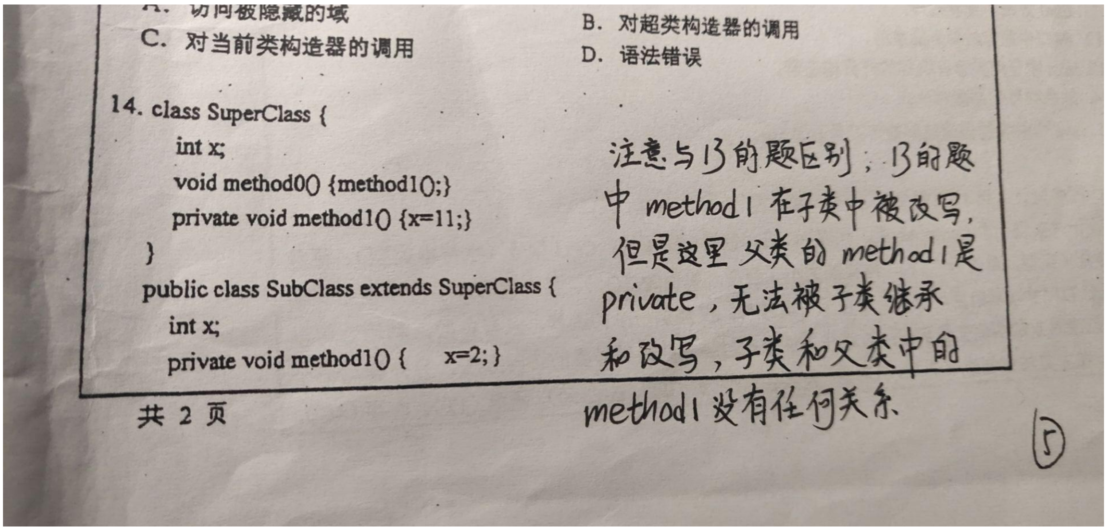
8. 下列代码哪行会出错？ C

```
 1)  public void modify() {
 2)    int I, j, k;
 3)    I=100;
 4)    while(I>0){
 5)      j=I*2;
 6)      System.out.println("The value of j is " +j);
 7)      k=k+1;
 8)      I—;
 9)    }
10) }
```

A. line 4　　B. line 6　　C. line 7　　D. line 8

局部变量无缺省值

使用局部变量前要先初始化



A. 访问被隐藏的域　　B. 对超类构造器的调用

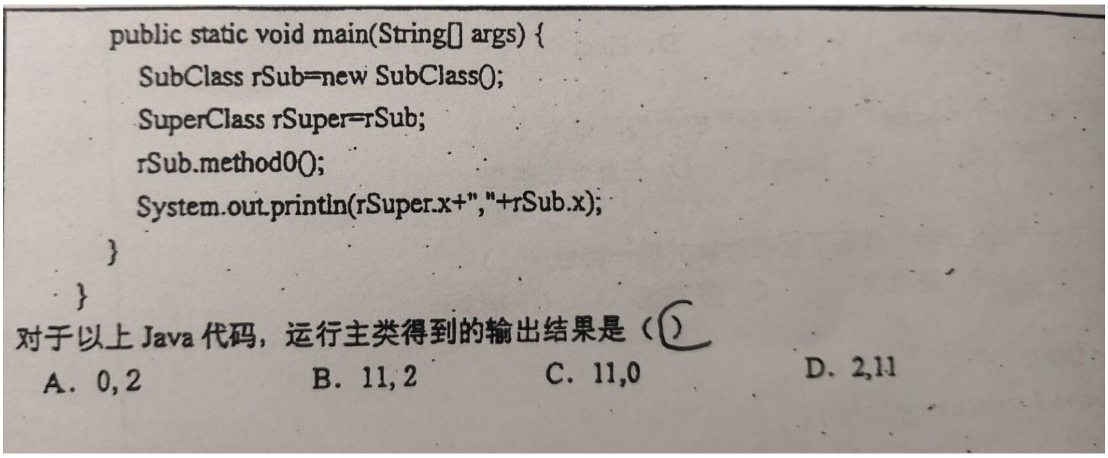
C. 对当前类构造器的调用　　D. 语法错误

```
14. class SuperClass {
      int x;
      void method0() {method1();}
      private void method1() {x=11;}
   }
   public class SubClass extends SuperClass {
     int x;
     private void method1() {    x=2; }
```

注意与13的题区别，13的题中 method1 在子类中被改写，但是这里 父类的 method1是 private，无法被子类继承和改写，子类和父类中的 method1 没有任何关系

共 2 页

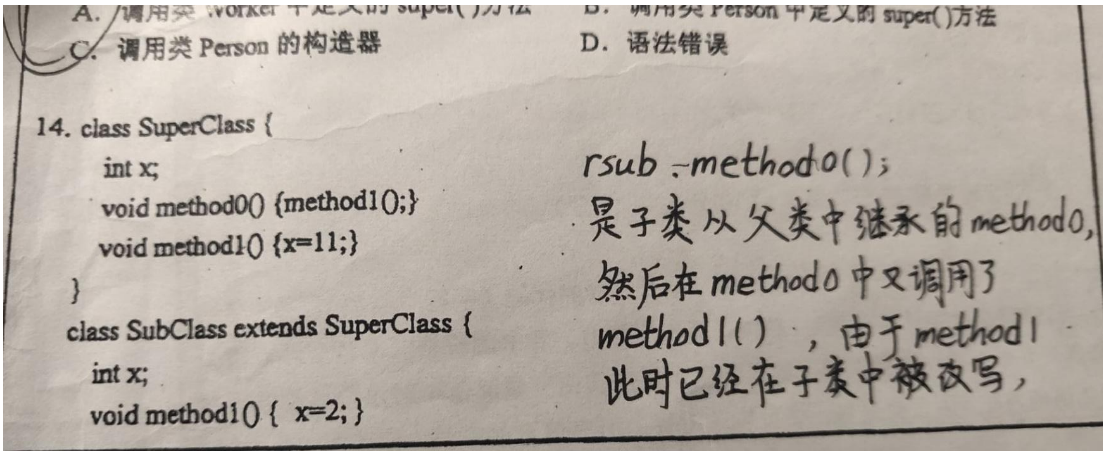
⑤



```
    public static void main(String[] args) {
      SubClass rSub=new SubClass();
      SuperClass rSuper=rSub;
      rSub.method0();
      System.out.println(rSuper.x+","+rSub.x);
    }
 }
```

对于以上 Java 代码，运行主类得到的输出结果是（C）

A. 0, 2　　B. 11, 2　　C. 11,0　　D. 2,11



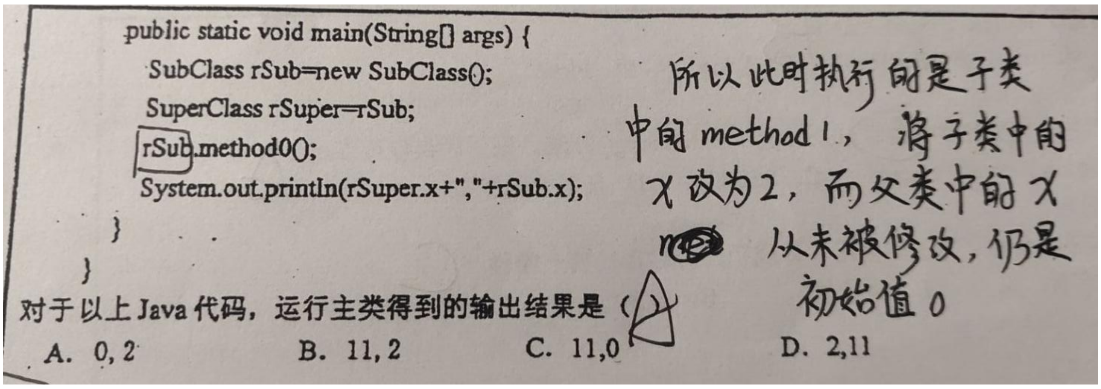
A. 调用类 Worker 中定义的 super( )方法　　B. 调用类 Person 中定义的 super( )方法

[✓ C. 调用类 Person 的构造器]　　D. 语法错误

```
14. class SuperClass {
      int x;
      void method0() {method1();}
      void method1() {x=11;}
   }
   class SubClass extends SuperClass {
     int x;
     void method1() {  x=2; }
```

rsub.method0();

是子类从父类中继承的method0，然后在method0中又调用了method1()，由于method1此时已经在子类中被改写，



```
    public static void main(String[] args) {
      SubClass rSub=new SubClass();
      SuperClass rSuper=rSub;
      rSub.method0();
      System.out.println(rSuper.x+","+rSub.x);
    }
  }
```

所以此时执行的是子类中的 method1，将子类中的x改为2，而父类中的x 从未被修改，仍是初始值0

对于以上 Java 代码，运行主类得到的输出结果是（A）

A. 0, 2　　B. 11, 2　　C. 11,0　　D. 2,11