第三部分　　JAVA 代码查错

1.

```
abstract class Name {
    private String name;
    public abstract boolean isStupidName(String name) {}
}
```

大侠们，这有何错误?

答案: 错。abstract method 必须以分号结尾，且不带花括号。

2.

```
public class Something {
    void doSomething () {
        private String s = "";
        int l = s.length();
    }
}
```

有错吗?

答案: 错。局部变量前不能放置任何访问修饰符 (private，public，和 protected)。final 可以用来修饰局部变量
(final 如同 abstract 和 strictfp，都是非访问修饰符，strictfp 只能修饰 class 和 method 而非 variable)。

3.

```
abstract class Something {
    private abstract String doSomething ();
}
```

这好像没什么错吧?

答案: 错。abstract 的 methods 不能以 private 修饰。abstract 的 methods 就是让子类 implement(实现)具体细节的，怎么可以用 private 把 abstract
method 封锁起来呢? (同理，abstract method 前不能加 final)。

4.

```
public class Something {
    public int addOne(final int x) {
        return ++x;
    }
}
```

这个比较明显。

答案: 错。int x 被修饰成 final，意味着 x 不能在 addOne method 中被修改。

5.

```
public class Something {
    public static void main(String[] args) {
        Other o = new Other();
        new Something().addOne(o);
    }
```

```
    public void addOne(final Other o) {
        o.i++;
    }
}
class Other {
    public int i;
}
```

和上面的很相似，都是关于 final 的问题，这有错吗?

答案: 正确。在 addOne method 中，参数 o 被修饰成 final。如果在 addOne method 里我们修改了 o 的 reference (比如: o = new Other();)，那么如同上例这题也是错的。但这里修改的是 o 的 member vairable (成员变量)，而 o 的 reference 并没有改变。

6.

```
class Something {
    int i;
    public void doSomething() {
        System.out.println("i = " + i);
    }
}
```

有什么错呢? 看不出来啊。

答案: 正确。输出的是"i = 0"。int i 属於 instant variable (实例变量，或叫成员变量)。instant variable 有 default value。int 的 default value 是 0。

7.

```
class Something {
    final int i;
    public void doSomething() {
        System.out.println("i = " + i);
    }
}
```

和上面一题只有一个地方不同，就是多了一个 final。这难道就错了吗?

答案: 错。final int i 是个 final 的 instant variable (实例变量，或叫成员变量)。final 的 instant variable 没有 default value，必须在 constructor (构造器)结束之前被赋予一个明确的值。可以修改为"final int i = 0;"。

8.

```
public class Something {
    public static void main(String[] args) {
        Something s = new Something();
        System.out.println("s.doSomething() returns " + doSomething());
    }
    public String doSomething() {
        return "Do something ...";
    }
}
```

看上去很完美。

答案: 错。看上去在 main 里 call doSomething 没有什么问题，毕竟两个 methods 都在同一个 class 里。但仔细看，main

是 static 的。static method 不能直接 call non-static methods。可改成"System.out.println("s.doSomething() returns " + s.doSomething());"。同理，static method 不能访问 non-static instant variable。

9.

此处，Something 类的文件名叫 OtherThing.java

```
class Something {
    private static void main(String[] something_to_do) {
        System.out.println("Do something ...");
    }
}
```

这个好像很明显。

答案: 正确。从来没有人说过 Java 的 Class 名字必须和其文件名相同。但 public class 的名字必须和文件名相同。

10.

```
interface   A{
    int x = 0;
}
class B{
    int x =1;
}
class C extends B implements A {
    public void pX(){
        System.out.println(x);
    }
    public static void main(String[] args) {
        new C().pX();
    }
}
```

答案：错误。在编译时会发生错误(错误描述不同的 JVM 有不同的信息，意思就是未明确的 x 调用，两个 x 都匹配（就象在同时 import java.util 和 java.sql 两个包时直接声明 Date 一样）。对于父类的变量,可以用 super.x 来明确，而接口的属性默认隐含为 public static final.所以可以通过 A.x 来明确。

11.

```
interface Playable {
    void play();
}
interface Bounceable {
    void play();
}
interface Rollable extends Playable, Bounceable {
    Ball ball = new Ball("PingPang");
}
class Ball implements Rollable {
    private String name;
    public String getName() {
```

```
        return name;
    }
    public Ball(String name) {
        this.name = name;
    }
  public void play() {
        ball = new Ball("Football");
        System.out.println(ball.getName());
    }
}
```

这个错误不容易发现。

答案: 错。"interface Rollable extends Playable, Bounceable"没有问题。interface 可继承多个 interfaces，所以这里没错。问题出在 interface Rollable 里的"Ball ball = new Ball("PingPang");"。任何在 interface 里声明的 interface variable (接口变量，也可称成员变量)，默认为 public static final。也就是说"Ball ball = new Ball("PingPang");"实际上是"public static final Ball ball = new Ball("PingPang");"。在 Ball 类的 Play()方法中，"ball = new Ball("Football");"改变了 ball 的 reference，而这里的 ball 来自 Rollable interface，Rollable interface 里的 ball 是 public static final 的，final 的 object 是不能被改变 reference 的。因此编译器将在"ball = new Ball("Football");"这里显示有错