

7. Java调试案例讲解

1. 什么是调试？

当我们发现程序中存在的问题的时候，我们去**查找和改正错误的过程称为调试（debugging）**。

调试的一般途径是采用各种方法逐步**缩小** bug 所在的范围，最终**定位bug**的准确位置。

2. 常见的调试方式

2.1 阅读代码跟踪程序问题

- 适用于代码量短小
- 常见的方式是逐行阅读代码和插入一些打印语句观察代码的问题

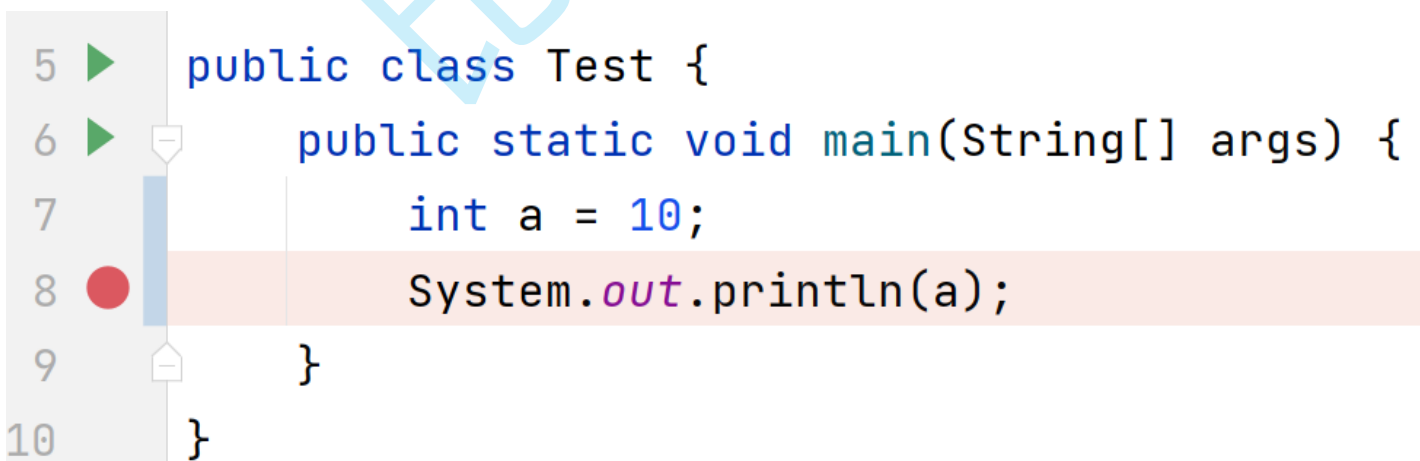
2.2 使用工具

- 适用于复杂程序
- 使用集成开发工具中集成的调试器

3. 使用IDEA调试Java程序

3.1 如何打断点

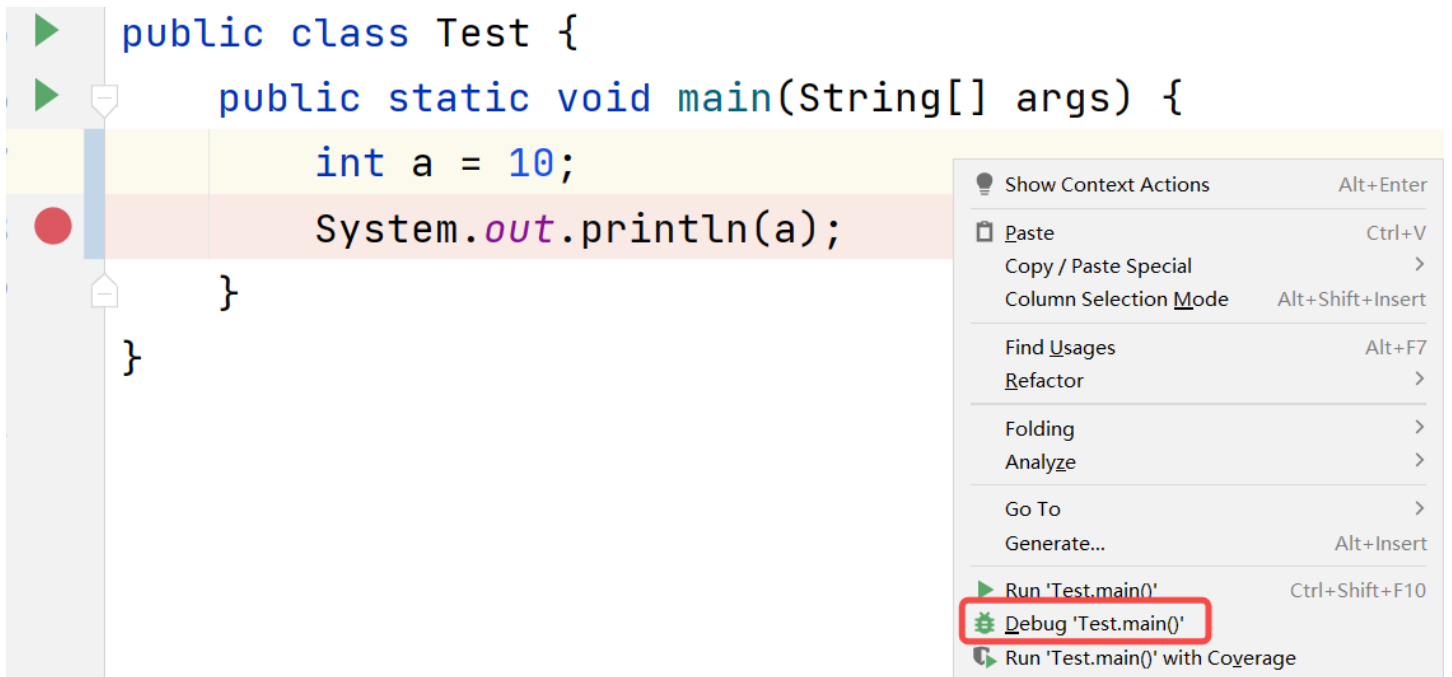
点击鼠标左键，出现红色按钮



```
5  ▶ public class Test {  
6  ▶     public static void main(String[] args) {  
7      int a = 10;  
8  ●     System.out.println(a);  
9      }  
10 }
```

3.2 如何启动调试

鼠标右键，点击Debug...



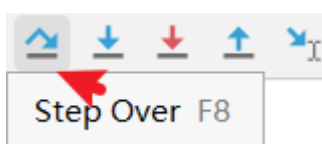
3.3 调试界面介绍



3.4 逐过程

逐行执行，不进入方法内部

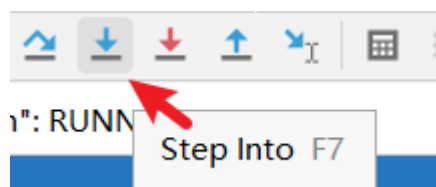
快捷键：Step Over (F8)



3.5 逐语句

进入方法内部执行

快捷键：Step Into (F7)



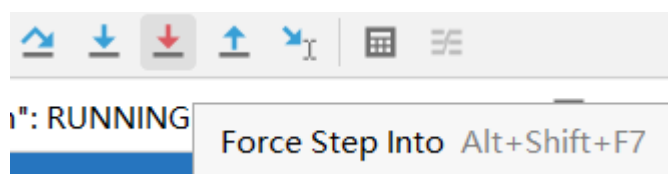
3.6 强制步入功能

Force Step Into 是一种强制步入功能，它允许调试器进入任何方法，包括那些通常会被 Step Into 功能跳过的方法。

与普通 Step Into 的区别：

- 普通 Step Into (F7) 通常会跳过库方法、无源代码的方法等。
- Force Step Into 会进入任何方法，包括那些通常被跳过的方法。

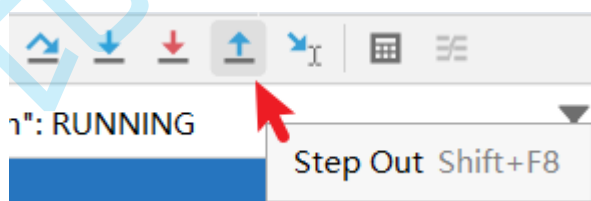
快捷键：Alt + Shift + F7



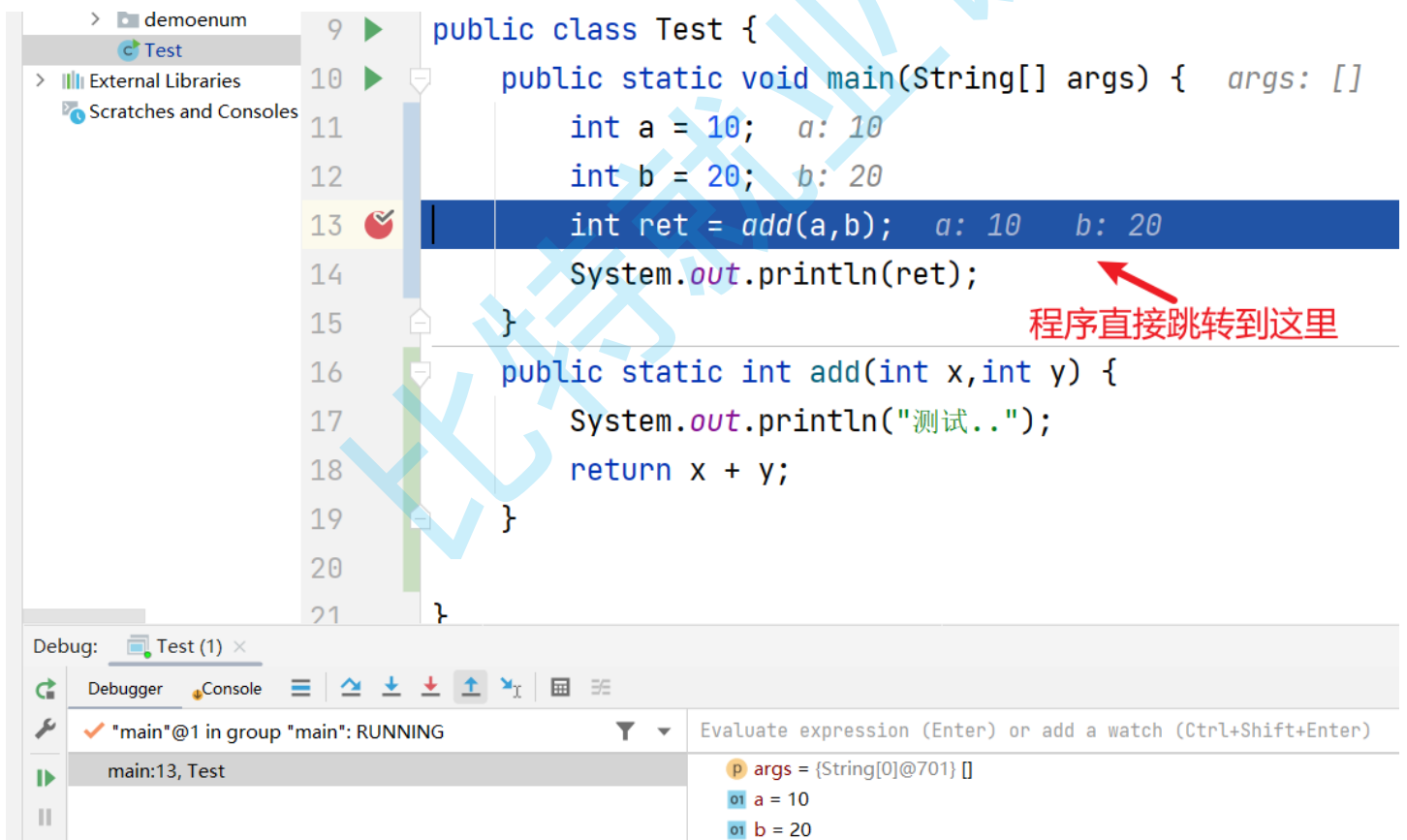
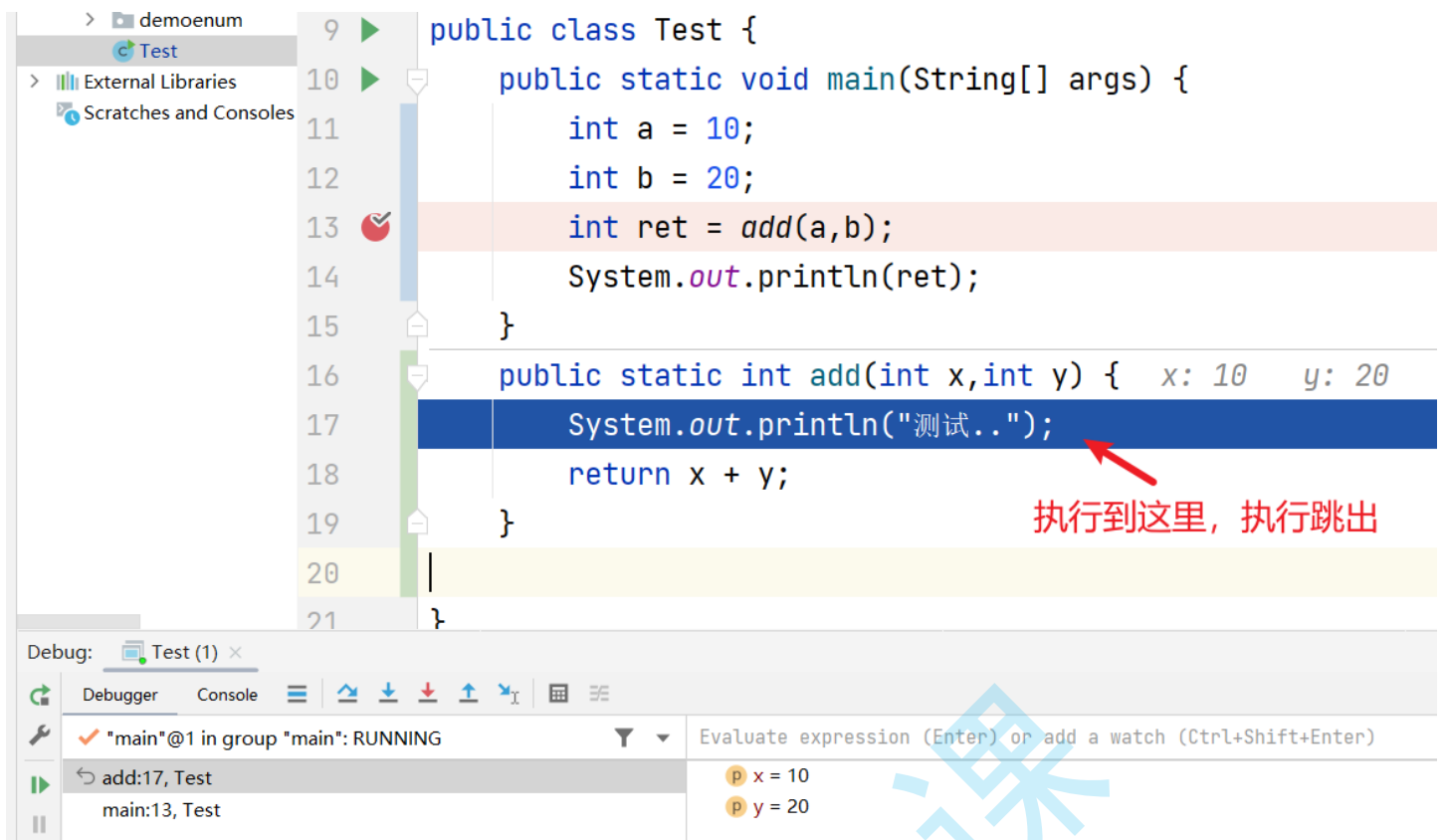
3.7 跳出功能

跳出当前方法，它让程序执行完当前方法的剩余部分，然后返回到调用该方法的位置。快速完成当前方法的执行。

快捷键：Step Out (Shift + F8)



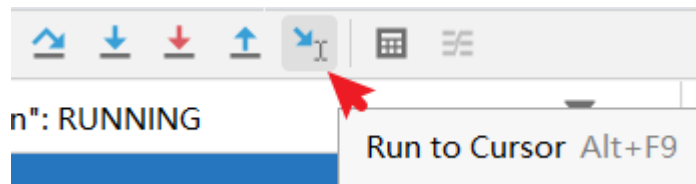
假设进入了一个方法，此时需要跳出，不执行后续方法。



3.8 运行到光标所在位置

运行到光标所在位置

快捷键: alt+F9



比如示例：

```
public class Test {
    public static void main(String[] args) {    args: []
        int a = 10;
        int b = 20;
        int ret = add(a,b);
        System.out.println(ret);
    }
    public static int add(int x,int y) {
        System.out.println("测试..");
        return x + y;
    }
}
```

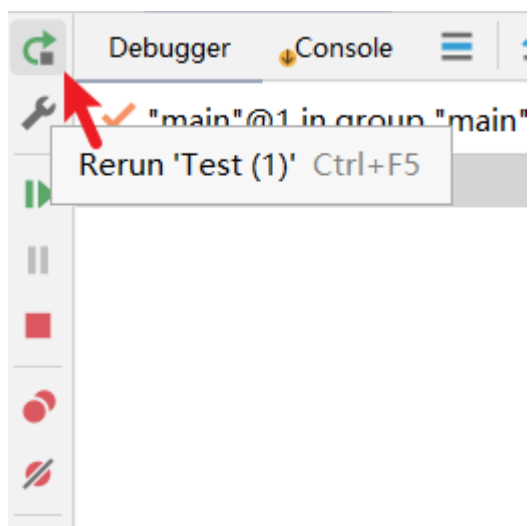
光标所在位置

程序跳转到光标处

```
9 public class Test {
10 public static void main(String[] args) {    args: []
11 int a = 10;    a: 10
12 int b = 20;    b: 20
13 int ret = add(a,b);    a: 10    b: 20    ret: 30
14 System.out.println(ret);    ret: 30
15 }
16 public static int add(int x,int y) {
17 System.out.println("测试..");
18 return x + y;
19 }
20 }
```

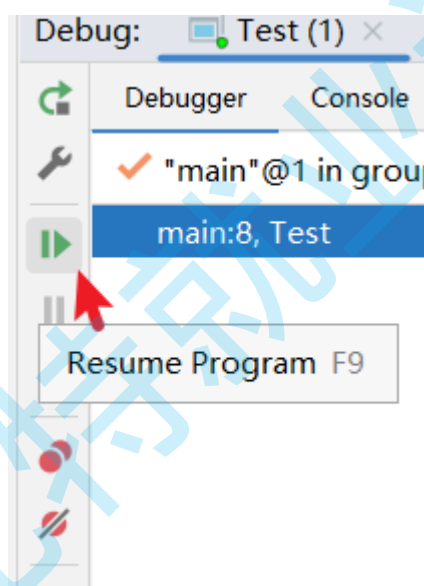
3.9 重新调试

快捷键：ctr+F5



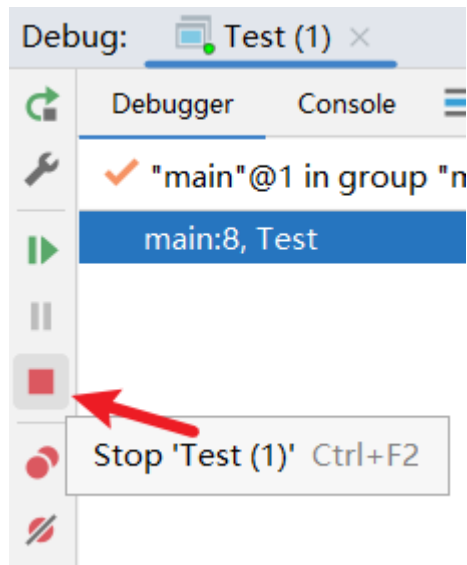
3.10 跳到下一个断点

从当前断点处，直接跳转到下一个断点处



3.11 暂停调试

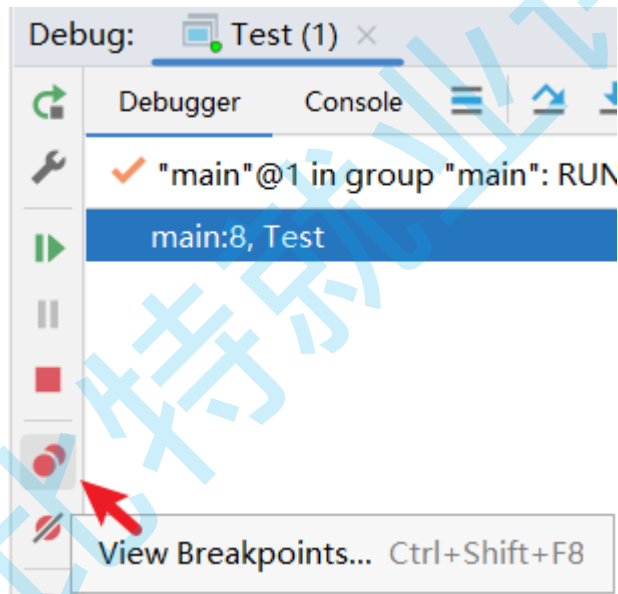
快捷键：ctr+F2



3.12 显示所有断点

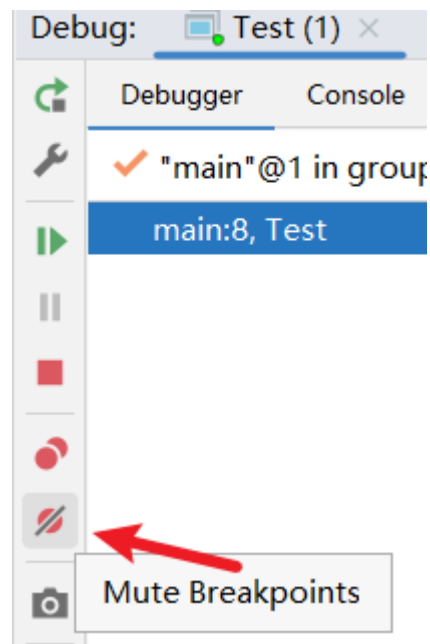
是一个集中管理所有断点的界面，可以查看、编辑、启用/禁用或删除断点。

快捷键：通常是 Ctrl + Shift + F8



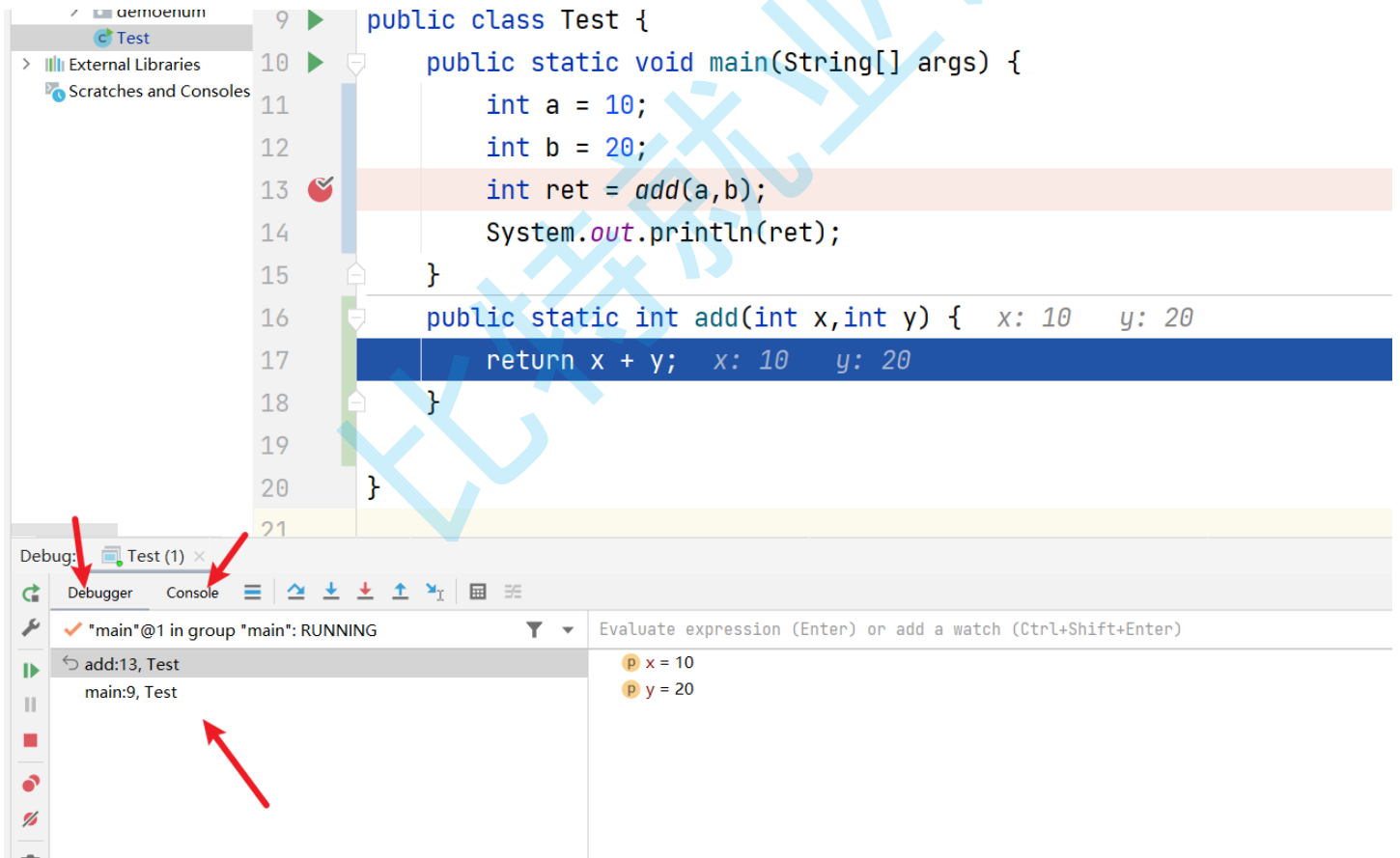
3.13 屏蔽断点

临时禁用所有断点的功能，而不需要逐个禁用或删除它们



4. 其他

4.1 Debugger标签下会展示调用堆栈以及变量等表达式的值



此时add:13 main:9 属于调用堆栈信息，先调用main方法，后调用add方法

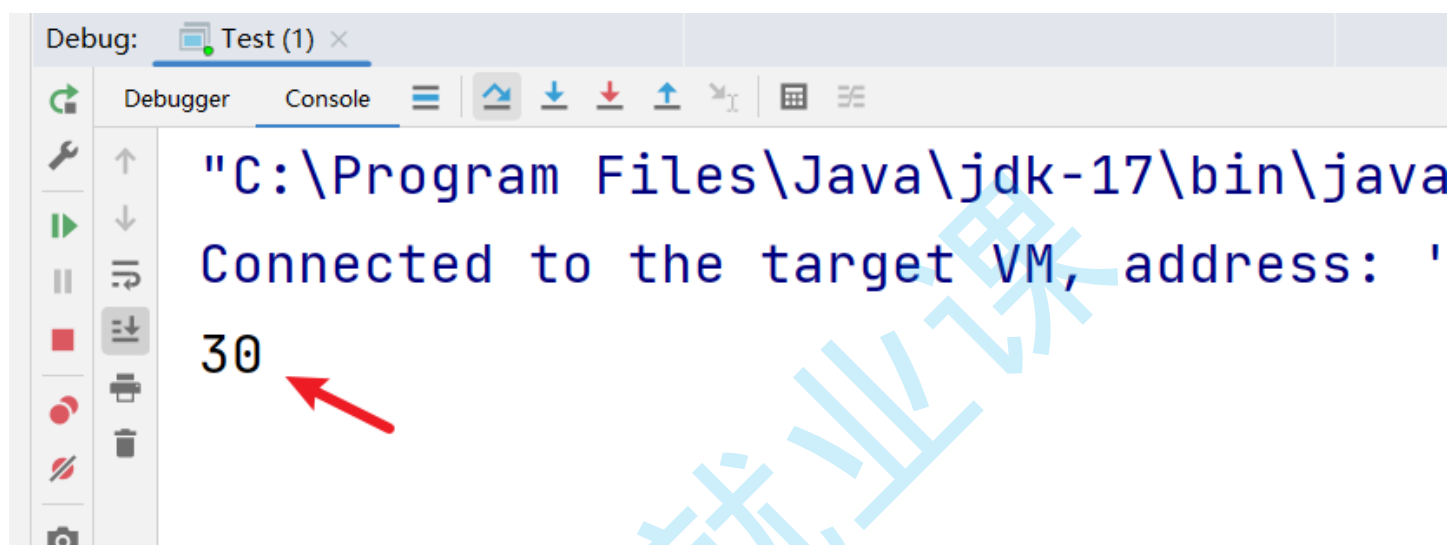
标有 "Evaluate expression (Enter) or add a watch", 这里可以输入表达式进行求值或添加到监视列表，如：


```
add(10,20)*2  
result = 60  
x = 10  
y = 20
```

可以输出表达式，按下回车计算结果

4.2 Console标签下会展示控制台的输入输出信息

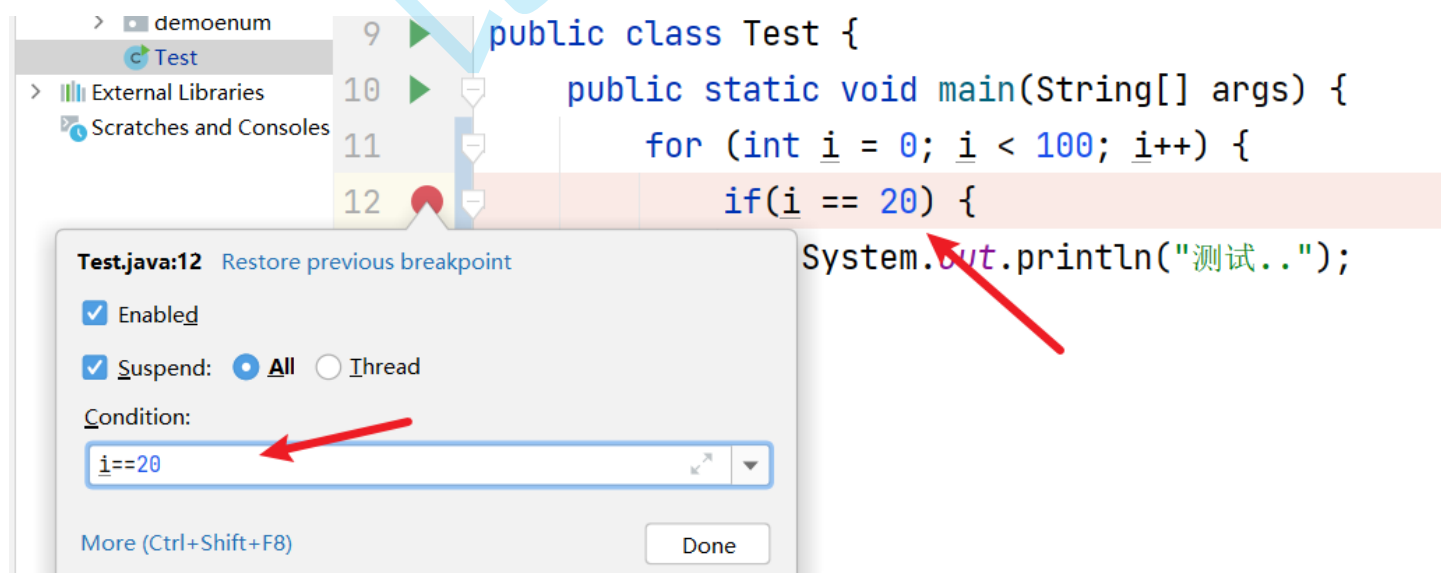
只要遇到输出语句，此时就可以查看输出的结果



4.3 条件断点

当一个代码需要满足某种条件进行调试，此时可以使用调试断点

- 打断点
- 鼠标右键，添加条件即可



- 开始调试程序会停在满足条件的情况下

```
9  ▶ public class Test {
0  ▶     public static void main(String[] args) {    args: []
1      for (int i = 0; i < 100; i++) {    i: 20
2      if(i == 20 = true) {    i: 20
3          System.out.println("测试..");
4      }
5  }
6  }
7  }
```

Debugger toolbar:

Debugger status: n: RUNNING

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

Variables:

- p args = {String[0]@699} []
- 01 i = 20 ←

完