

目标：

通过本次活动结束时，你应该能够做到以下几点：

- Ø 获得进一步的理解 *如果* 和 *如果别的* 的声明
- Ø 了解的基础知识 *而* 语句 (又名 *while* 循环)
- Ø 推出 `ArrayList` 类 (`java.util.ArrayList` 中)

描述：

在此活动中，您将创建一个 **NumberOperations** 类将持有一个整数值，并提供方法，以该值执行各种操作。您还将下载和修改一个名为类 **NumberOpsDriver** 它创建一个 `ArrayList` 对象，然后在从键盘的值读入。当值不为零，该方案为价值创造的 `NumberOperations` 类的实例，增加了该实例到 `ArrayList`，然后从键盘读取下一个值。

路线：

第1部分：**NumberOperations**：方法存根 (又名框架代码)

不要忘记添加 `Javadoc` 注释: 类，构造函数和每个方法都需要一个。

- 创建一个类调用 `NumberOperations` 与所谓的实例变量 数类型 `INT`。

```
public class NumberOperations {
    private int number;
}
```

- 添加方法 存根 以下方法。前两个给出; 做你自己休息。

Ø 构造函数称为 `numberIn` `int` 参数

```
public NumberOperations(int numberIn) {
}
```

Ø 的 `getValue`：不带任何参数; 返回一个 `int` 值

```
public int getValue()
{
    return 0; // placeholder return
}
```

创建占位回报对自己的每个方法的方法存根。

Ø `oddsUnder`：不带任何参数; 返回一个字符串

Ø `powersTwoUnder`：不带任何参数; 返回一个字符串

Ø 更伟大：采用称为 `compareNumber` `int` 参数; 返回一个 `int`

Ø 的 `toString`：不带任何参数; 返回一个字符串

编译 `NumberOperations` 并运行在交互以下。不要继续下去，直到你的程序编译和下面的代码运行没有交互运行时错误。请注意，返回值将在“存根”方法的占位符值。

```
NumberOperations numOps = new NumberOperations ( 5 );
字符串S1 = numOps.oddsUnder ( );
字符串s2 = numOps.powersTwoUnder ( );
INT N1 = numOps.isGreater ( 2 );
串S3 = numOps.toString ( );
```

第2部分：NumberOperations：构造函数，的getValue和toString

- 在您的构造函数中添加代码，将设定的值 数 至 `numberIn`。
- 在你的getValue方法，删除占位符返回，返回的值 数。
- 替换用下面的代码toString方法占位符的回报：

```
return number + "";
```

[需要注意的是串联数，它是一个int，用空字符串的结果为一个字符串。]

编译NumberOperations并运行在交互面板下面的代码。 不要继续下去，直到下面的代码中的互动而不会出现错误运行。

```
// NumberOperations numOps = 新 NumberOperations ( 5 );
// numOps.getValue ( )
||| 五
// numOps //显示了toString返回值
||| 五 |||
```

第3部分：NumberOperations：oddsUnder方法 - 返回包含正奇数小于数的值的字符串。

- 创建oddsUnder一个局部变量叫 产量 并将其初始化为空字符串常量。

```
public String oddsUnder() {
    String output = "";
}
```

- 添加本地int变量 `i` 和while循环，将通过循环，只要重复的值 `i` 小于的值 数。

```
int i = 0;
while (i < number) {

}
```

- 里面的上述循环的，添加代码，将concatenate的价值 `i` 如果它是一个奇数。同时增加的价值 `i` 循环的每次迭代期间。

```
if(i % 2 != 0) {
    output += i + "\t";
}
i++;
```

- 在循环后，添加代码 返回输出的值。

编译NumberOperations并运行在交互面板下面的代码。 不要继续下去，直到下面的代码中的互动而不会出现错误运行。

```
// NumberOperations numOps = 新 NumberOperations ( 9 );
// numOps.oddsUnder ( )
||| 1 3 5 7
```

部4A：NumberOperations：powersTwoUnder方法 - 返回包含两个小于数的值的正功率的字符串。

- 创建一个名为在powersTwoUnder一个局部字符串变量 产量 并将其初始化为空字符串文字，你在做oddsUnder。
- 创建一个名为类型为int的另一个局部变量 权力 并将其值初始化为1。
- 添加while循环将通过每个号码重复，直到价值 数。

```
while (powers < number) {
```

- 在while循环中，添加代码，将concatenate权力输出的值，如果它是2的幂，然后计算2的下一个动力（下面的评论是可选的）。

```
output += powers + "\t"; // concatenate to output
powers = powers * 2; // get next power of 2
```

- 在循环后，添加代码 返回输出的值。

编译NumberOperations并运行在交互面板下面的代码。 不要继续下去，直到下面的代码中的互动而不会出现错误运行。

```
NumberOperations numOps = new NumberOperations ( 20 );
numOps.powersTwoUnder ( )
```

```
1      2      4      8      16      III
```

4B部分：NumberOperations：isGreater方法

- 删除从isGreater占位符返回和补充的代码，将返回1，如果数目大于compareNumber更大，-1，如果数目大于compareNumber是更小，或0，如果数量相等。

```
if (number ____ compareNumber) {
    return 1;
}
else if (number ____ compareNumber) {
    return -1;
}
else {
    return 0;
}
```

编译NumberOperations并运行在交互面板下面的代码。 不要继续下去，直到下面的代码中的互动而不会出现错误运行。

```
NumberOperations numOps = new NumberOperations ( 10 );
numOps.isGreater ( 2 )
```

```
1
```

```
numOps.isGreater ( 15 )
```

```
- 1
```

```
numOps.isGreater ( 10 )
```

```
0 一世
```

部分5A：NumberOpsDriver类

- 下载所谓的驱动程序 *NumberOpsDriver.java* 然后添加所指示的代码（在//注释中描述），使得它执行以下操作：提示用户通过一个空格，接着数字0分离数（整数）的列表；读取列表中的第一个号码；进入循环并且当数目不为0，创建一个对象NumberOperations，加NumberOperations反对称为一个ArrayList *numOpsList*，然后阅读该列表中的下一个号码（即，通过循环迭代）。一旦0值从列表中读取，循环终止。现在，使用while循环的第二，打印出每个NumberOperations在其沿着ArrayList对象“赔率下”和它的“下的2的幂”。示例输出：



```

MM«中号 — jGRASP EXEC : JAVA NumberOpsDriver
MM$M 输入一个空格，然后分离0的正整数的列表：
%%$M 12 9 17 0
MM$M 适用于：12
MM$M      1 3 5 7 9 11：下赔率
MM$M      2下的权力：                1 2 4 8
MM$M 适用于：9
MM$M      赔率下：1 3 5 7
MM$M      2下的权力：                1 2 4 8
MM$M 适用于：17
MM$M      1 3 5 7 9 11 13 15：下赔率
MM$M      2下的权力：                1 2 4 8 16
MM$MMM«中号 — jGRASP：操作完成。

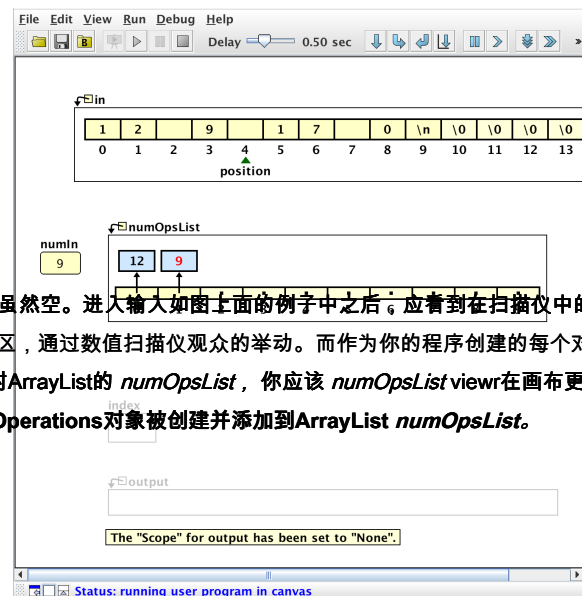
```

部分5B：乳宁NumberOpsDriver在画布

- 下载jGRASP画布文件
NumberOpsDriver.jgrasp_canvas.xml 并在文件夹中保存该活动的源文件。

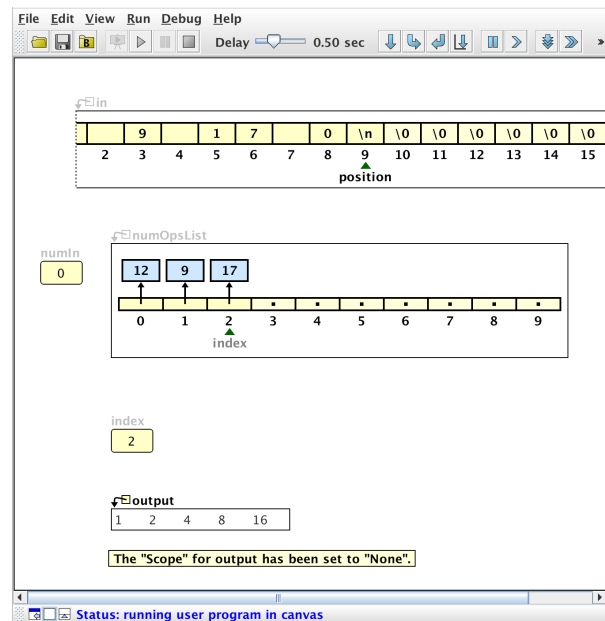
- 现在点击画布上运行按钮  上
桌面工具栏。画布文件打开后，点击播放按钮  你呢

应该看到扫描仪观众 在 和ArrayList *numOpsList* 变得活跃，虽然空。进入输入如图上面的例子中之后，应看到在扫描仪中的值在观众。由于每个值是由你的程序读取你应该看到它的缓冲区，通过数值扫描仪观众的举动。而作为你的程序创建的每个对象NumberOperations的价值，并增加了NumberOperations反对ArrayList的 *numOpsList*，你应该 *numOpsList* viewr在画布更新。右边的图显示了 后两个值的帆布已经被读取和两个NumberOperations对象被创建并添加到ArrayList *numOpsList*。



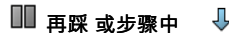
- 作为该计划发挥你应该看到观众 产量 这是在oddsUnder () 和powersTwoUnder () 方法的局部变量。请注意，“范围”为 产量

已经通过使用浏览器下拉菜单并选择设置为“无”范围 接着没有。这允许观看者显示的值 范围 无论哪个局部变量的是。右边的图显示了画布程序执行powersUnder () 方法 numOpsList的第三元件上，同时（即，在索引2的元件）。



显示低于17两个数字的五点力。

- 在画布上运行时，你可以暂停



再踩 或步骤中



如

适当的查看所关注行为。您也可以停止



画布，然后在画布上运行



和播放



再次。为规范

在画布程序的速度，降低或增加使用延迟滑块步骤之间的延迟

Delay 0.30 sec.

- 你应该在画布上运行此程序进行试验，直到你确信你了解双方的 *NumberOperations* 和 *NumberOpsDriver* 类。由于本次活动介绍 *同时声明*，你应该通过每个while语句的停顿画布，然后单步，以确保您了解他们。在主方法中，第一while语句中的值读出，第二个填充NumberOperations对象的ArrayList的。你也应该通过每个而单步循环中oddsUnder () 和powersTwoUnder () 方法。

网络-CAT - 您在下一节中创建的jGRASP项目后，您应该提交您的

NumberOperations.java 和 NumberOpsDriver.java 通过项目的工具栏上的Web-CAT按钮文件的Web-CAT。