Consider the following two procedures:

```java
static void linyur(int wig, int k) {
    int biz = wig - k;
    out.print(wig + 2 * k - biz);
    out.print(' ');
}

public static void main() {
    int zit = 3;
    linyur(6, zit);
    linyur(zit - 1, 4);
}
```

These introduce two related new concepts:

1. **Local variables**
2. **Parameter variables**
Consider the procedure named linyur:

```java
static void linyur(int wig, int k) {
    int biz = wig - k;
    out.print(wig + 2 * k - biz);
    out.print(' ');
}
```

The red boxes show where some variables are declared

Remember that `int` means "create a variable with an integer value".

The variables are named `wig`, `k`, and `biz`. They each hold an integer value.

They are called **LOCAL** variables - because they are declared **INSIDE** a procedure.

This means they **ONLY EXIST** during a **SINGLE ACTIVATION** of the procedure.
Local variables are divided into two kinds:

(1) Parameter variables

(2) Working variables

Parameter variables are declared in the parentheses pair ( ) immediately following the procedure name.

Working variables are declared inside the braces pair { } which provide the one sequence command that is the body of the procedure.

In Java, a working variable is usually initialized as part of its declaration like this:

\[
\text{int biz = 23*5;}
\]

But (in Java) parameter variables are initialized in the caller !!!!
Consider the procedure named linyur:

```java
static void linyur(int wig, int k) {
    int biz = wig - k;
    out.print(wig + 2 * k - biz);
    out.print(' ');
}
```

Thus linyur has three local variables wig, k, and biz. Of these, wig and k are parameter variables and biz is a working variable.
Consider the procedure named main:

```java
public static void main() {
    int zit = 3;
    linyur(6, zit);
    linyur(zit-1, 4);
}
```

Thus **main** has one **local** variable **zit**. There are no **parameter** variables and **zit** is the only **working** variable.

**Side Note:** the main used here is irregular and is missing its usual parameter variable named **args** - this is done to simply some of the homeworks.
A local context is a group of variables.

(1) Each time a procedure is activated, a new local context is created for it.

(2) At the end of that activation, the corresponding local context is DESTROYED.

(3) In a local context for a procedure are ALL its local variables for ONE activation.

A local context will be diagrammed in a table - for example:

<table>
<thead>
<tr>
<th>Name of local variable</th>
<th>Value (history) of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>wig</td>
<td>4</td>
</tr>
<tr>
<td>k</td>
<td>19 21 28</td>
</tr>
<tr>
<td>biz</td>
<td>-7</td>
</tr>
</tbody>
</table>

Each local context will have an identification number.
A Procedure Activation Chart (PAC) shows all the activations of the procedures in an execution of the program.

So, when those procedures have Local Contexts - the PAC is a convenient place to record the activity in each such Context.

All the Local Contexts are kept in a column to the side of the PAC and the initial horizontal line segment of an activation is labeled with the identification number of the Local Context created at that time.

For now, you MUST assume that a procedure CAN ONLY see and use its OWN local variables - the ones in the Local Context that corresponds to that procedure's activation.

This will become clearer in the examples in subsequent notes.