Local variables

Local variable: A variable declared in a method body.

```
// Swap x and z.
int temp= x;
x= z;
z= temp;
```

Form of declaration:

<type> <variable-name> ;

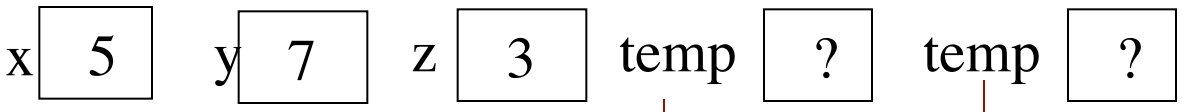
or

<type> <variable-name> = *<expression>* ;

1. When a local variable is created and destroyed.
2. The scope of a local variable.
3. Guidelines for naming a local variable.

Creation & destruction of local variables

call: m(5, 7, 3)



All pars & local variables created, and arg values stored in pars, before execution of method body

```
/** = smallest of x, y, and z */
public void m(int x, int y, int z)     {
    if (x > y) {
        // Swap x and y.
        int temp;
        temp= x;   x= y;   y= temp;
    }
    if (x > y) {
        // Swap x and z.
        int temp;
        temp= x;   x= z;   z= temp;
    }
    return x;
}
```

Exist as long as body is executed

Destroyed when execution terminates

Scope of local variables

call: p(5, 7)

Scope of local variable: from just after its declaration to end of block in which it is declared.

```
/** = sum of values in range m..n.
    Precondition: m <= n+1. */
public void p(int m, int n)  {
    int s= m;        k= 10;  illegal
    int k;
    // inv: s = sum of m..k–1
    for  (k= m; k <= n; k= k+1) {
        s= s + k;
    }

    return s;
}
```

Scope of s

Scope of k

Scope of for-loop counter

call: p(5, 7)

Scope of local variable: from just after its declaration to end of block in which it is declared.

```
/** = sum of values in range m..n.
    Precondition: m <= n+1. */
public void p(int m, int n)  {
    int s= m;
    // inv: s = sum of m..k–1
    for  (int k= m; k <= n; k= k+1) {
        s= s + k;
    }
    k= 10;  illegal
    return s;
}
```

Scope of k

Local-variable names

```
/** = sum of values in range m..n.
    Precondition: m <= n+1. */
public void p(int m, int n)  {
    int s= m;
    // inv: s = sum of m..k–1
    for  (int k= m; k <= n; k= k+1) {
        s= s + k;
    }

    return s;
}
```

Long parameter names complicate

```
/** = sum of values in range first_value..last_value.
    Precondition: first_value <= last_value+1. */
public void p(int first_value, int last_value)     {
    int s= first_value;
    // inv: s = sum of first_value..k–1
    for  (int k= first_value; k <= last_value;
                                          k= k+1) {
       s= s + k;
    }

    return s;
}
```

Short parameter names simplify — as do short local-variable names

```
/** = sum of values in range m..n.
    Precondition: m <= n+1. */
public void p(int m, int n)  {
    int s= m;
    // inv: s = sum of m..k–1
    for  (int k= m; k <= n; k= k+1) {
        s= s + k;
    }

    return s;
}
```

Short parameter names are better —as long as the specification mentions the parameters appropriately.

Short local-variables names are better —as long the local variables are appropriately described.