Behavior of parameter variables and local variables

- **Pre-requisite: from cs170/cs171**
    - **Lifetime of *Parameter* variable:**
        - A **parameter variable** is **created** (i.e., **reserve memory space** for the **paramter variable**) at the ***beginning*** of the **execution of the method invocation (call)**
        - A **parameter variable** is **destroyed** (i.e., **reserved memory space** for the **paramter variable** is **unreserved (freed)**) at the ***end (termination)*** of the **execution of the method invocation (call)**

Example:

```
void f(int a)
{  <--- variable a exists because f(a) has created a

   .....   location where variable a exists ....

}       <--- variable a is "destroyed"

void main(String[] args)
{

    ....

    f(a);     <---- variable a is created and then f() is invoked

    ....
}
```

- **Lifetime of *Local* variable:**
    - A **local variable** is **created** (i.e., **reserve memory space** for the **local variable**) at the ***place of definition*** of the **local variable**
    - A **local variable** is **destroyed** (i.e., **reserved memory space** for the **local variable** is **unreserved (freed)**) at the ***end (termination)*** of the **execution of the method invocation (call)**

Example:

```
void f(....)
{
   int x;        // variable x begins to exist

   x = 1;

   int y;        // variable y begins to exist

   y = 2;
}            <----- variables x and y are destroyed
```

- **Note:**

- The **lifetime** of a **parameter variable** is **identical** to a **local variable** that is **defined at the *beginning* (start)** of a **method**
- **Difference:**
  - **parameter variables** are **initialized** by the **caller method**
  - **Local variables *cannot* be initialized** by the **caller method**

- **Further pre-requisite from cs170/171**
  - **Important fact:**
    - **Parameter variables** and **local variables** are ***private*** to ***each*** **method *invocation***
    - In other words:
      - **Every time** a **method** is **invoked (called)**, a ***new* set** of **parameter variables *and* local variables** are **created (reserve memory)**

  - **Example:**

```
public class Behavior
{
   public static int count = 0;

   public static void f(int a)
   {
      int b;                  // Local variable

      b = a + 100;

      if ( a == 0 )
         return;
      else
         f(a-1);

      System.out.println(" a = " + a + "    b = " + b);
   }

   public static void main(String[] args)
   {
      f(3);
   }
}
```

Output:

```
a = 1    b = 101
a = 2    b = 102
a = 3    b = 103
```

How is the program executed:

```
main:      +-->  f(a = 3)      +--> f(a = 2)      +--> f(a = 1)      +--> f(a = 0)
          /                   /                  /                  /         |
    f(3) -       b = 100 + 3 /      b = 100 + 2 /      b = 100 + 1 /          | (return)
                            /                  /                  /           |
                 f(a-1) ----        f(a-1) ----        f(a-1) ----            |
                                                                              |
                                                       print(a,b) (1, 101) <--+
                                                           |
                                    print(a,b) (2, 102) <--+
                                        |
                 print(a,b) (3, 103) <--+
                    |
    <---------------+
```

```
                      ^                   ^                 ^
                      |                   |                 |
Parameter var:      a = 3               a = 2             a = 1
Local variable:     b = 103             b = 102           b = 101
```

- **Example Program:** (Demo above code)

  Example

  - Prog file: click here

- **Non-recursive methods**

  - **Non-recursive method:**

    - A **non-recursive method** is a **method** that will ***not* be invoked** if it is **currently *active***

  Example:

```
void f( ... )
{


}

void main( .... )
{

   ....

   f( ... );    // f is invoked and becomes active

         During the entire time that f() is ACTIVE
         the method f() will not be invoked again !
}
```

  - **Local (and parameters if you are careful) variables for *non-recursive* methods:**

    - **Local variables** for **non-recursive method** can be **reserved** using the **DS directive**

      (The **location** of the **local variables** is usually **after the** `rts` instruction for the method)

- This is **possible** because **only *one* invocation** will be **active**

  The variables defined by the **DS directive** is **adequate**

- ***Recursive* methods**
  - **Recursive method:**

    - A **recursive method** is a **method** that ***will* be invoked** when it is ***already* currently *active***

    Example:

```
public class Behavior
{
   public static int count = 0;

   /* ---------------------------------------
      A recursive method
      --------------------------------------- */
   public static void f(int a)
   {
      int b;                 // Local variable

      b = a + 100;

      if ( a == 0 )
         return;
      else
         f(a-1);

      System.out.println(" a = " + a + "    b = " + b);
   }

   public static void main(String[] args)
   {
      f(3);
   }
}
```

Notice that `f()` was invoked while `f()` is active:

```
main:      +-->  f(a = 3)      +-->  f(a = 2)      +-->  f(a = 1)      +-->  f(a = 0)
          /                   /                   /                   /         |
    f(3) -       b = 100 + 3 /      b = 100 + 2 /      b = 100 + 1 /          |  (return)
                            /                  /                  /           |
                 f(a-1) ----        f(a-1) ----        f(a-1) ----            |
                                                                              |
                                                       print(a,b)  (1, 101) <--+
                                                           |
                                    print(a,b)  (2, 102) <--+
                                        |
                 print(a,b)  (3, 103) <--+
                     |
    <----------------+
```

```
                      ^                   ^                 ^
                      |                   |                 |
Parameter var:      a = 3               a = 2             a = 1
Local variable:     b = 103             b = 102           b = 101
```

- **Local variables and parameters variables for *recursive* methods:**

  - **Local variables *and* parameter variables** for ***recursive* method *cannot*** be **reserved** using the **DS directive**

  - This is **impossible** because there are ***more than one* invocation** will be **active**

    Each invocation must uses a **different set** of **local variables** and **parameter variables**

    The **DS directive** can **only create (reserve space for)** ***one* set of variables**