Methods with arguments and return values

Recall: Parameter passing ...

```java
public static void main(String[] args) {
    for (int i=0; i < 10; ++i) {
        nStars(i);
        System.out.println();
    }

    for (int i=10; i > 0; --i) {
        nStars(i);
        System.out.println();
    }
}
```

Recall: Parameter passing

```java
/**
 * Print "*" n times.
 * @param n Number of stars to print
 */
static void nStars(int n) {
    for (int i=1; i <= n; ++i) {
        System.out.print("*");
    }
}
```

Result ...

```java
for (int i=0; i < 10; ++i) {
    nStars(i);
    System.out.println();
}
```

This value i

Is copied into n

```java
for (int i=1; i <= n; ++i) {
    System.out.print("*");
}
```

Not the same i.
A new variable with the same name!
The method call “stack”

```java
public static void main(String[] args) {
    for (int i=0; i < 10; ++i) {
        nStars(i);
        System.out.println();
    }
    for (int i=10; i > 0; --i) {
        nStars(i);
        System.out.println();
    }
}
```

```
static void nStars(int n) {
    for (int i=1; i <= n; ++i) {
        System.out.print("*");
    }
}
```

Java method signature

```
static int min (int t1, int t2) {
    if (t1 <= t2) {
        return t1;
    } else {
        return t2;
    }
}
```

Arguments (input)

```
static int min (int t1, int t2) {
    if (t1 <= t2) {
        return t1;
    } else {
        return t2;
    }
}
```

The inputs of this method are two integers.
ex:  int a =1; int b=2;
    int x = min(a,b);  // note names!

“Input” and “Output”

The input and output of a **program** are typically files, console, screen, etc.

The input and output of a **method** are its arguments (parameters) and result (return value)
Method return type (output)

static int min (int t1, int t2) {
    The output of this method is an int.
    I can write  int x = min(7, 9);
}

... then in main ...
int a =5; int b = 7;
int c = diff( b, a );

O, be some other name! – Juliet

static int diff (int a, int b) {
    return a – b;
}

... then in main ...
int a =5; int b = 7;
int c = diff( b, a );
    // ??? What happens

The called method gets its own copies of the inputs, by position (not by name).
“Pass by value”

```java
static void foo( int a, int b, double c, String d ) { ... }
```

The “actual arguments” x, y, 37.489, “et tu, Brute?” are copied into the “formal arguments” of the method. The copies become distinct, local variables.

---

What does it print?

```java
static void foo(int x, int y) {
    x = 17;
    y = 19;
}
...
int x = 3; int m = 22;
foo(x, m);
System.out.println(“Now “ + x + “ and “ + m);
```

---

procedures and functions

**static void foo( ... ) { ... }**

“void” type (procedure): Doesn’t return anything must do something (e.g., printing)

**static int foo( ... ) { .... }**

“int” type (function): Returns a value should usually be a “pure” function, no side effects

---

What makes a good method? (review)

Simplifies the code that calls it
Isolates a design decision (easier to change)
Used more than once
Can be tested separately

... 

*A good method may have only some of these properties. Few have all.*
Bad method smells

Complicated description
If the simplest description is “blah blah and blah and blah except blah or blah”, maybe it shouldn’t be a method

Have to keep looking back at it
I should be able to use the method without remembering details of how it works

Password character test

How will you organize the logic for “uses only characters of kinds x, y, z, and at least one of each”?

Start with pseudocode ... save the Java details for until you are convinced your logic is good.

Let’s write a method for the triangles program

Equilateral? Isosceles? Max of 3?

Let’s write “isValid” for the triangle inequality. Propose at least two approaches.