Methods, Libraries, Debugging, and Information Processing

Overview

Methods
- Functional decomposition
- Method syntax
- Passing and returning values
- The call stack
- Overloading

Libraries
- Calling methods from other classes
- Application programming interfaces (APIs)
- Packages and `import`
- JUnit

Debugging

Information Processing

Methods

Functional Decomposition

Why

Program understanding

Debugging

Code reuse

How

Design from top down

Implement from bottom up

Each method should do one job (return a value or have a side effect)

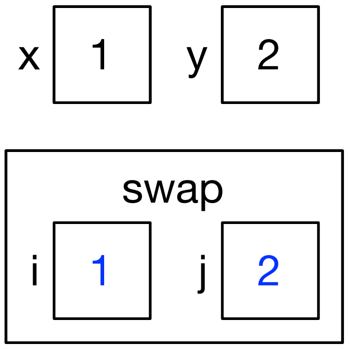
Method syntax

```
/** Returns true if word is in dictionary. */
public static boolean contains(String word, String[] dictionary) {
    for (int i = 0; i < dictionary.length; i++) {
        if (word.equals(dictionary[i])) {
            return true;
        }
    }
    return false;
}
```

Passing and returning values

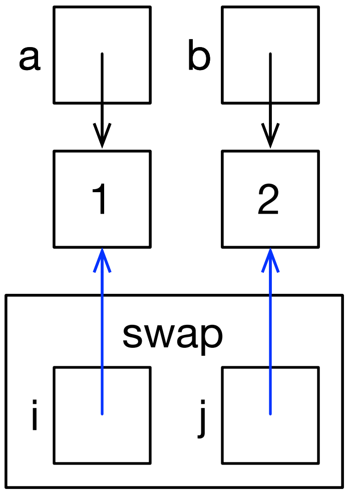
When you pass an argument to a method or return a value from a method, you send a copy …

```
int x = 1;
int y = 2;
// This can't be done:
swap(x, y);
```



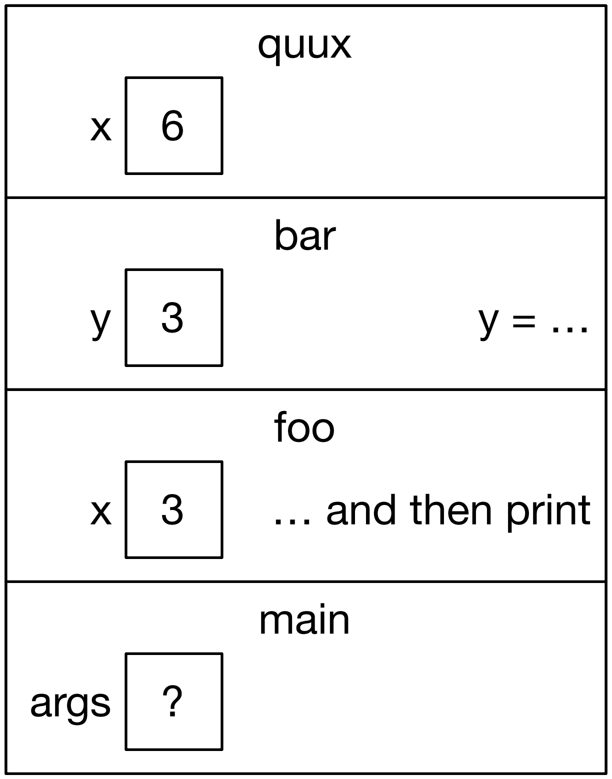
… but if what you're copying is a pointer, the thing on the other end of the pointer isn't copied!

```
int[] a = {1};
int[] b = {2};
// This can:
swap(a, b);
```



The call stack

```
public class Stacktacular {

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

    public static void foo(int x) {
        bar(x);
        StdOut.println(x);
    }

    public static void bar(int y) {
        y = quux(y * 2);
    }

    public static int quux(int x) {
        return x + 1;
    }

}
```



Overloading

```
public static int size(int n) {
   return n;
}

public static int size(String s) {
   return s.length();
}

public static int size(double[] a) {
   return a.length;
}
```

Libraries

Calling methods from other classes

ClassName **.** *methodName* (*arguments*)

Application programming interfaces (APIs)

See the book’s website.

Google for built-in Java classes like String.

Generate your own with Javadoc.

Packages and `import`

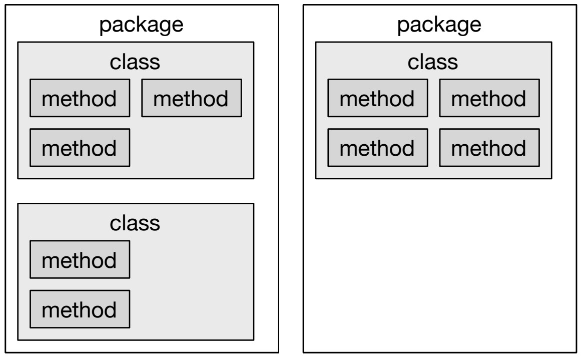
```
java.awt.Color tan = new java.awt.Color(210, 180, 140);
```

or

```
import java.awt.Color;
```

before class begins, then:

```
Color tan = new Color(210, 180, 140);
```



You can't import from the default package.

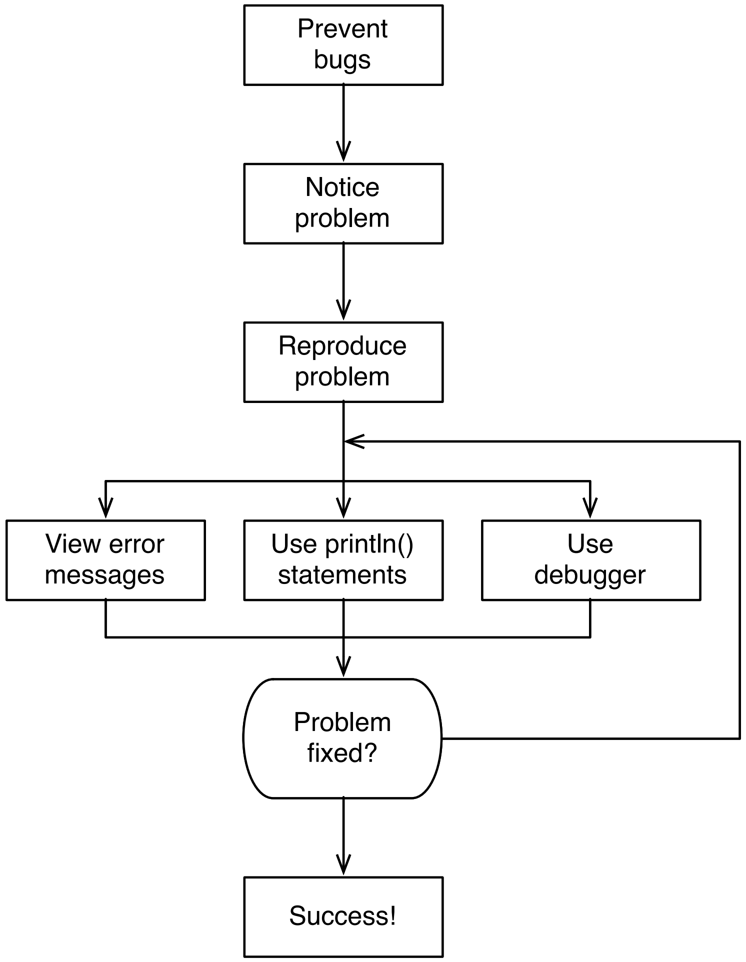
JUnit

http://screencast.com/t/svxwr8LR

http://screencast.com/t/PgA21Udwl

Debugging

http://screencast.com/t/NEgEMW6sNB2



Information processing

Skim, read, and read closely, as appropriate

What do those symbols mean?

What information is implicit in the diagram?

What other resources are available?

Review

Methods copy values, but not things on the other end of pointers.

The call stack keeps track of work left to do.

Methods can be overloaded.

You can call methods from other classes and even other packages.

Test first with JUnit.

Debug methodically.

Information processing is a skill.