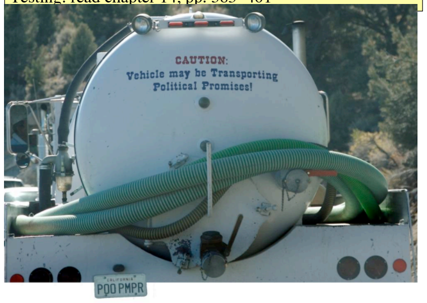
CS100J 8 March 2007 Executing method calls. Testing

Drawing frames for method calls: read pp. 93-94

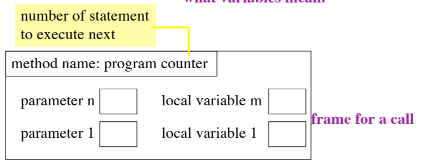
Testing: read chapter 14, pp. 385–401



1

Executing method calls, pp 93-94

Understanding this not only prepares you for prelim 2, it helps you understand how recursion can work and how a method determines what variables mean.



Execution of method call
1. Draw a frame for the call.
2. Assignment arg values to the pars
3. Execute method body
4. Erase frame, and, for a function, return value of the **return** expression.

In step 3, look in frame for variables/methods.

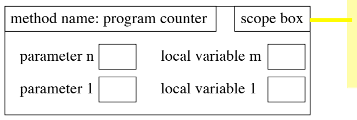
2

Executing method calls, pp 93-94. The scope box

the scope box indicates where to look for a name that is not in the frame. It contains:

static method: name of class.

non-static method: name of object in which the called mehod resides.



Execution of method call
1. Draw a frame for the call.
2. Assignment arg values to the pars
3. Execute method body
4. Erase frame, and, for a function, return value of the **return** expression.

In step 3, look in frame for variables/methods. If not there, look in place given by scope box.

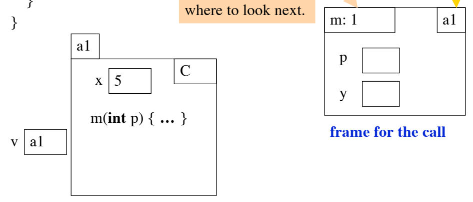
3

Executing method calls, pp 93-94

```
public class C {
  int x;
  public int m(int p) {
    int y= p + x;
    return y;
  }
}
```

When executing method body, look in frame for variables/methods. If not there, use scope box to tell where to look next.

non-static method: name of object



v.m(6) evaluate this expression

4

Testing: Read chapter 14.

Bug: Error in a program.

Testing: Process of analyzing, running program, looking for bugs.

Test case: A set of input values, together with the expected output.

Debugging: Process of finding a bug and removing it.

Exceptions: When an error occurs, like divide by 0, or s,,charAt[I] when I = – 1, Java *throws an exception*. A lot —generally too much— information is provided.

5

Exceptions: When an error occurs, like divide by 0, or s.charAt[i] when i = – 1, Java *throws an exception*.

```
06 /** = String s truncated …. */
07   public static String truncate5(String s) {
08       int b= 10 / 0;
09       if (s.length() <= 5)
10           return s;
11       return s.substring(0,5);
12   }
```

Turn on line numbering in DrJava. Preferences / Display Options

important part

call stack

```
ArithmeticException: / by zero
  at A4Methods.truncate5(A4Methods.java:8)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(….java:39)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(….java:25)
  at java.lang.reflect.Method.invoke(Method.java:585)
```

6

Debugging a program

When an error occurs, you have to play detective and find it. That process is called debugging. The place where the bug is may be far removed from the place where an error is revealed.

Simplest way for you to debug at this point is to put print statements, suitably annotated, at judiciously chosen places in the program.

Debugging a program

When an error occurs, play detective and find it. Called debugging. The place where the bug is may be far removed from the place where an error is revealed.

```
public static HSV RGB2HSV(Color rgb) {
  …
  /**Figure out MAX and MIN*
  double MAX= 0; double MIN=  0;
  if (R>G && R>B)     {MAX= R; }
  if (G>B && G>R)    {MAX= G;}
  if (B>R && B>G)    {MAX= B;}
  if (R<G && R<B)    {MIN= R; }
  if (G<B && G<R)    {MIN= G; }
  if (B<R && B<G)    {MIN= B;}

  System.out.println("R " + R + ", G " + G +
                     ", B ” + B + ", MAX " + MAX);
```

If you just output the numbers without naming them, you will have trouble.

```
public static HSV RGB2HSV(Color rgb) {
  …
  if (R>G && R>B)     {MAX= R; }
  if (G>B && G>R)    {MAX= G;}
  if (B>R && B>G)    {MAX= B;}
  if (R<G && R<B)    {MIN= R; }
  if (G<B && G<R)    {MIN= G; }
  if (B<R && B<G)    {MIN= B;}
  System.out.println("R " + R + ", G " + G +
                  ", B ” + B + ", MAX " + MAX);
```

call and output

```
> A4Methods.RGB2HSV(new java.awt.Color(255,255,128))
R 1.0, G 1.0, B 0.5019607843137255, MAX 0.0
```

Look! MAX is 0 and not 1! if conditions should be >= , not >

```
public static HSV RGB2HSV(Color rgb) {
  …
  double MAX= Math.max(R, Math.max(G, B));
  double MIN= Math.min(R, Math.min(G, B));
```

The above is a better way to calculate the MAX and the MIN. It is better to rely on what has been programmed before than to duplicate the work. It saves time and leads to fewer errors.