Understanding the execution of

- local variable declaration (in a method body)
- new expression (3 steps)
- method call (method frames, call stack)

Examples from previous exams
- code execution (Q4 from 2008 fall final, modified)
- method call (Q3 from 2007 fall final)

Important!
- All previous finals included some questions about code execution
- You need to know how to draw variables, objects, method frames...
- The purpose of such questions on executing statements with new expressions and method calls is to test your understanding of how Java programs are executed

Code segment (in a method body)

int a = 3;
C x = new C(a);
C y = new C(a);
x = y;

The first thing to do?

to draw all local variables

Code segment (in a method body)

private int f;
C(int k) { f = k; }
}

int a = 3;
C x = new C(a);
C y = new C(a);
x = y;

Execution of new expression

- 3 steps for executing the new expression
  - create a new folder (object) of the class with a unique name (place it in the class file drawer)
  - initialize the fields in the object by executing the constructor
  - yield the name of the object as the value of the new expression
public class C {
    private int f;
    C(int k) { f = k; }
}

int a = 3;
C x = new C(a);
C y = new C(a);
x = y;

When is the local variable b inside the loop created?
It is created once with other local variables when we start executing the method (in the method frame).
Not after the loop starts.

Execute the call:
Store.session();

1: Item one = new Item("ipod", 20);
2: Item two = new Item("wii", 32);
3: Item treat = two;
4: Item three = one;
5: three.add(4);
6: System.out.println(one);
7: System.out.println("Cost of Item: " +
    Item.getTotalCost());
8: System.out.println("Are they the same? " +
    (one.getName() == treat.getName()));
9: System.out.println("Are they the same? " +
    one.getName().equals(treat.getName()));
10: System.out.println("Are they the same? " +
    (one.getName() == three.getName()));

answers:
6: "ipod:24"
7: "Cost of Item: 56"
8: "Are they the same? false"
9: "Are they the same? false"
10: "Are they the same? true"

The frame (the box) for a method call

Remember: Every method is in a folder (object) or in a file-drawer.

method name: instruction counter scope box

local variables (don’t deal with these now)
parameters

Draw the parameters as variables.

number of the statement of method body to execute NEXT. Helps you keep track of what statement to execute next. Start off with 1.
The frame (the box) for a method call:

Remember: Every method is in a folder (object) or in a file-drawer.

- method name: instruction counter
- local variables (don't deal with these now)
- scope box
- parameters

If this is a static method, the method should be in the file-drawer, thus, the scope box should contain the class name. If it is not static, it is in the folder (Object), the scope box should contain the object.

Scope of local variable: the sequence of statements following it within the containing "block".

To execute the call setScore(100):

1. Draw a frame for the call.
2. Assign the value of the argument to the parameter (in the frame).
3. Execute the method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)
4. Erase the frame for the call.

Exercise 1

Call Stack

Call Stack is the stack of frames for uncompleted method calls. A frame for a method call lasts as long as the method call is being executed. When the call is finished, the frame is erased.

This fact explains why local variables do not retain their values from one call of a method to the next call of the same method. All the information about the first call is in a frame, and the frame is erased when the call is completed.

Question 2 (10 points) Executing method calls. Suppose the program contains the method call function(int x, int y) { int temp; x = y; y = temp; return x; }

Execute the method call

int temp = function(10, 20);

Answer:

- The method is called.
- Function name: function
- Parameters: int x, int y
- Method call: function(10, 20)
- Arguments: 10, 20
- Return value: 20
- Frame is erased
- The program continues with temp = 20.
Step Into vs Step Over

**Step Into:**
- Draw the method call and execute the method call.

**Step Over:**
- Assume the function is doing exactly what it should do based on the specifications of the function.

Reverse: 1
- VectorTools
- 0
- 1
- 2
- a1

Reverse: 2
- VectorTools
- 0
- 1
- 2
- a1

Step Into:
- Draw the method call and execute the method call.

Step Over:
- Assume the function is doing exactly what it should do based on the specifications of the function.