Biggest issue!!! You can't do questions on this topic correctly unless you draw variables, draw objects when they are created, and draw frames for method calls. Learning to do this will help you to do the same thing when trying to find errors in your programs.

Understanding 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

```java
public class C {
    private int f;
    public C(int k) { f = k; }
}
```

The first thing to do? draw all local variables

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

Evaluation of new expression

- 3 steps in evaluating the new expression `new C(args)`
  - create a new folder (object) of class C with a unique name (place it in the class file drawer)
  - Execute the constructor call `C(args)`
  - yield the name of the object as the value of the new expression
**Code Execution (Q4 from 2008 fall final, modified)**

Execute the call: `Store.session();`

```java
public class Store {
    public static void session() {
        Item one = new Item("ipod", 20);
        Item two = new Item("wii", 32);
        Item treat = two;
        Item three = one;
        three.add(4);
        System.out.println(one);
        System.out.println("Cost of Item: " + Item.getTotalCost());
        System.out.println("Are they the same? " + (one.getName() == treat.getName()));
        System.out.println("Are they the same? " + one.getName().equals(treat.getName()));
        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"

**Code Execution (Q4 from 2008 fall final, modified)**

Execute the call: `Store.session();`

```java
public class Store {
    public static void session() {
        Item one = new Item("ipod", 20);
        Item two = new Item("wii", 32);
        Item treat = two;
        Item three = one;
        three.add(4);
        System.out.println(one);
        System.out.println("Cost of Item: " + Item.getTotalCost());
        System.out.println("Are they the same? " + (one.getName() == treat.getName()));
        System.out.println("Are they the same? " + one.getName().equals(treat.getName()));
        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
- **local variables (don’t deal with these now)**
- **scope box**

If this is a static method, this method is in the file-drawer, so the scope box contains the class name, if it is not static, it is in the folder (object), scope box contains the name of the object.

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

```java
/** = the max of x and y */
public static int max(int x, int y) {
    // Swap x and y to put the max in x
    if (x < y) {
        int temp;   // scope of temp
        temp = x;   // You can’t use temp down here
        x = y;
        y = temp;   // This is an error.
    }
    return x;
}
```

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.

Exercise 1

Exercise 1

1. Draw a frame for the call.
2. Assign arguments to the parameters (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.

**To execute the call:**

```
setScore: i
no
```

```
score
0
```

```
Score
```

**Score of local variable:** the sequence of statements following it.

```java
/** s contains a name in the form exemplified by “David Gries”.
Return the corresponding String “Gries, David”.
There may be 1 or more blanks between the names. */
public static String switchFormat(String s) {
    // Store the first name in variable f and remove f from s
deleation int k;   // Index of the first blank in s
    String f;   // The first name in s.
    f = s.substring(k);
    s = s.substring(k);   // Remove the blanks from s
    s = s.trim();
    return s + ”, ” + f + ”;” scope of f
}
```

Question 2 (5 points) Executing method calls. Suppose you have two public static methods: countFactors(x) returns the number of factors of x, and countPrimes(x) returns the number of primes less than x. In the following code:

```java
public class VowelTest {  // class VowelTest
    public static void main(String[] args) {
        int alphabet = 101;
        int value = 0;
        int value2 = 50;
        int value3 = 50;
        value = value + value2 + value3;
        System.out.println("Value: "+ value);
    }
}
```
**Step Into vs Step Over**

Step Into:
Draw the frame for the 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.