**Method Calls**

Consult relevant section in [JVM Spec](http://lara.epfl.ch/w/compilation:jvm_spec)

**Relevant Instructions**

Invoking methods:

* **invokestatic**
* **invokevirtual**

Returning value from methods:

* **ireturn**
* **areturn**
* **return**

**Translation Rule for Calls**

[[ x = myMethodName(e1,e2) ]] =

[[e1]]

[[e2]]

invoke(virtual/static) #13

istore #x

....

constant pool area

0: "hello, world"

1:

...

13: className.myMethodName/(II)I

...

**In Lower-Level Code**

Function calls are implemented by storing return addresses in a register (and saving the previous address on e.g. stack). Returning from a function is done by jumping to the value in the return register.

Stack is used to store local variables, as well as the operand stack.

Parameters are passed by pushing them onto the stack, and return result is also returned on the stack.

Higher-order functions are implemented by passing around function pointers and jumping to the address given by this function pointer.

Dynamic dispatch refers to the fact that the dynamic type of the object stored in a reference variable determines which method will be invoked. Therefore, objects behave as if they were storing function pointers. They can also be implemented in this way: a pointer to an object contains both the values of object fields and the vtable, which stores pointers to actual methods. What is usually known at compile time (at least in the case of single inheritance) is the offset of the method (the compiler knows at compile time whether the method called is 3rd or 5th among the methods stored in the object).