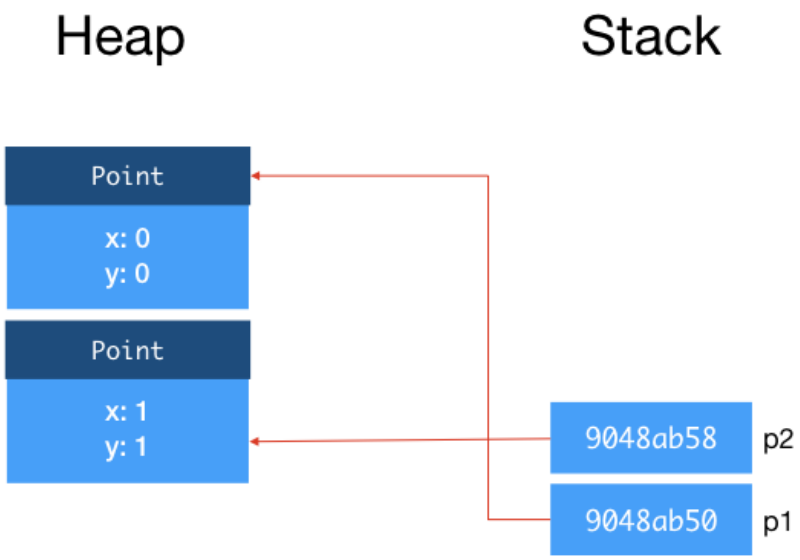
Stack and Heap Diagram Extended

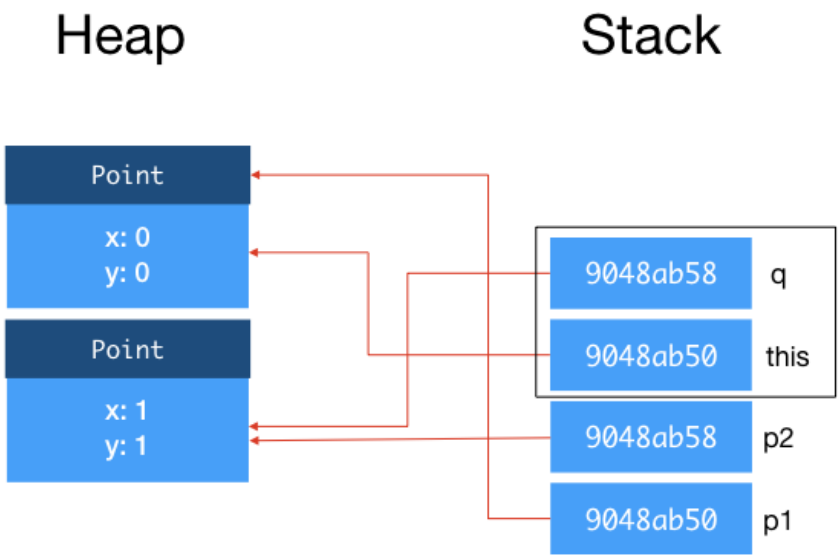
Method call



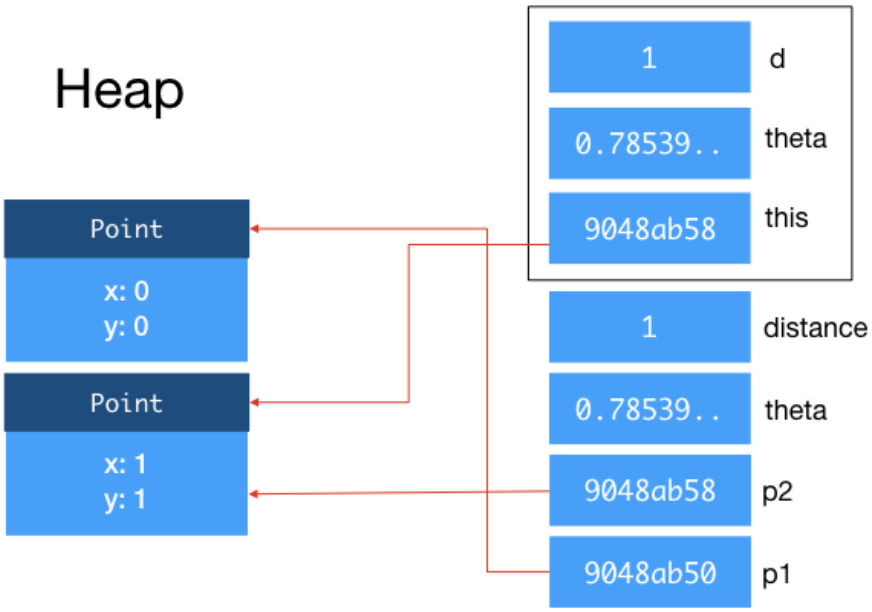
Before

JVM creates a *stack frame* for this instance method call. This stack frame contains + `this` reference. + The method arguments. `q` + Local variables within the method. *(Not shown)*

When a class method is called, the stack frame does not contain the ***`this`*** *reference.*



After



With Primitives

Note that `d` *and* `theta` *do not point to an object but instead are passed by value.*
Variable capture

Consider the program below:

```
class B {
  void f() {
    int x = 0;
    class A { // This is a local class
      int y = 0;
      A() {y = x + 1;}
    }
  A a = new A();
  }
}
```

Suppose that a variable b is an instance of class B, and a program calls b.f(). Sketch the content of the stack and heap immediately after the Line A a = new A() is executed. Label the values and variables / fields clearly. You can assume b is already on the heap and you can ignore all other content of the stack and the heap before b.f() is called.

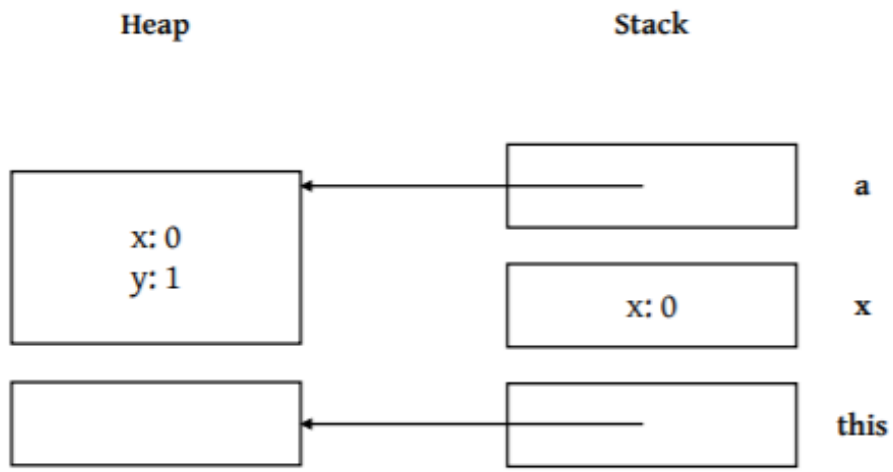


Figure 1: Answer

Variable capture: Local class makes a copy of local variables used from the enclosing method to within itself This stack frame contains (due to method call):

- `this` reference.
- No method arguments.
- Local variables within the method. `x`

This stack frame contains (due to variable declaration): + `a`, the variable initialised with a `new A()` object.

This heap contains (due to variable capture): + An instance of class `A`. + Captured variable `x` now part of its instance attributes. + Declared variable `y` now part of its instance attributes.