```
New Array:
index: 0 stack:  [] memory: {}
{'offset': 0, 'opr': 'push', 'value': {'type': 'integer', 'value': 3}}
index: 1 stack:  [3] memory: {}
{'offset': 1, 'opr': 'newarray', 'dim': 1, 'type': 'int'}
index: 2 stack:  [3, []] memory: {}
{'offset': 3, 'opr': 'dup', 'words': 1}
index: 3 stack:  [3, []] memory: {}
{'offset': 4, 'opr': 'push', 'value': {'type': 'integer', 'value': 0}}
index: 4 stack:  [3, [], 0] memory: {}
{'offset': 5, 'opr': 'push', 'value': {'type': 'integer', 'value': 1}}
index: 5 stack:  [3, [], 0, 1] memory: {}
{'offset': 6, 'opr': 'array_store', 'type': 'int'}
index: 6 stack:  [3, [1]] memory: {}
{'offset': 7, 'opr': 'dup', 'words': 1}
index: 7 stack:  [3, [1]] memory: {}
{'offset': 8, 'opr': 'push', 'value': {'type': 'integer', 'value': 1}}
index: 8 stack:  [3, [1], 1] memory: {}
{'offset': 9, 'opr': 'push', 'value': {'type': 'integer', 'value': 2}}
index: 9 stack:  [3, [1], 1, 2] memory: {}
{'offset': 10, 'opr': 'array_store', 'type': 'int'}
index: 10 stack:  [3, [1, 2]] memory: {}
{'offset': 11, 'opr': 'dup', 'words': 1}
index: 11 stack:  [3, [1, 2]] memory: {}
{'offset': 12, 'opr': 'push', 'value': {'type': 'integer', 'value': 2}}
index: 12 stack:  [3, [1, 2], 2] memory: {}
{'offset': 13, 'opr': 'push', 'value': {'type': 'integer', 'value': 3}}
index: 13 stack:  [3, [1, 2], 2, 3] memory: {}
{'offset': 14, 'opr': 'array_store', 'type': 'int'}
index: 14 stack:  [3, [1, 2, 3]] memory: {}
{'offset': 15, 'opr': 'store', 'type': 'ref', 'index': 0}
index: 15 stack:  [3] memory: {0: [1, 2, 3]}
{'offset': 16, 'opr': 'load', 'type': 'ref', 'index': 0}
index: 16 stack:  [3, [1, 2, 3]] memory: {0: [1, 2, 3]}
{'offset': 17, 'opr': 'push', 'value': {'type': 'integer', 'value': 0}}
index: 17 stack:  [3, [1, 2, 3], 0] memory: {0: [1, 2, 3]}
{'offset': 18, 'opr': 'array_load', 'type': 'int'}
index: 18 stack:  [3, 1] memory: {0: [1, 2, 3]}
{'offset': 19, 'opr': 'return', 'type': 'int'}
```

https://github.com/immarianaas/pa-23/tree/master/assignment-4

Which was the hardest to implement?

From what we managed to implement, the "invoke" operation.

Which operations did we purposefully not implement?

Purposefully, none.

But we couldn't implement some, such as invoking a function from the Java standard library. The "new" operation (and therefore, "throw" as well) is also not implemented correctly.

```
case "invoke":
    to_invoke = byteObj["method"]
    num_args = len(to_invoke["args"])
    args = stack[-num_args:]
    stack = stack[:-num_args]
    res = interpretMethod(
        to_invoke["ref"]["name"], to_invoke["name"], dict(enumerate(args))
    )
    stack.append(res)
```

Calls.java	
helloWorld	fib
👎	👎

Simple.java						
noop	zero	hundredAndTwo	identity	add	min	factorial
👍	👍	👍	👍	👍	👍	👍

Array.java								
first	firstSafe	access	**newArray**	newArrayOutOfBounds	accessSafe	bubbleSort	aWierdOneOutOfBounds	aWierdOneWithinBounds
👍	🤔	👍	👍	🤔	👎	👎	👍	🤔

Arianna, Kajsa, Kári, Mariana