

# THE COMMAND PATTERN IN ACTION: FUNCTIONAL CONSTRUCTS IN AN OBJECT-ORIENTED LANGUAGE

# PROGRAMMING PARADIGMS

OLD-SCHOOL LANGUAGES LIKE C

## IMPERATIVE PROGRAMMING

VARIABLES, LISTS, DICTIONARIES, FOR-LOOPS

## FUNCTIONAL PROGRAMMING

FUNCTIONS CALLING FUNCTIONS CALLING FUNCTIONS

## OBJECT-ORIENTED PROGRAMMING

CLASSES DEFINE INTERFACES; OBJECTS  
INSTANTIATE CLASSES; OBJECTS HAVE  
STATE (MEMBERS) AND BEHAVIOUR (METHODS)

EXCEL (YEP IN MANY WAYS  
EXCEL IS A PROGRAMMING LANGUAGE  
TOO! JUST NO FOR LOOPS)

JAVA, C++, C#

THE LINES BETWEEN THESE PROGRAMMING PARADIGMS  
CAN SOMETIMES GET A BIT BLURRY

CONSIDER A CLASS  
THAT HAS JUST 1  
METHOD

IT ENCAPSULATES ALL  
THE STATE NEEDED  
FOR THAT ONE METHOD  
TO DO ITS THING

**"ANONYMOUS CLASSES ARE AN EXCELLENT  
WAY TO ENCAPSULATE LITTLE BITS OF BEHAVIOR  
INTO OBJECTS"**

IT TURNS OUT A VERY HIGH PROPORTION OF  
ANONYMOUS CLASSES SIMPLY CONSISTED  
OF -

**OBJECTS THAT IMPLEMENT AN INTERFACE  
WITH JUST ONE FUNCTION**

## OBJECTS THAT IMPLEMENT AN INTERFACE WITH JUST ONE FUNCTION

SUCH INTERFACES ARE CALLED  
"FUNCTIONAL INTERFACES" OR  
"SINGLE ABSTRACT METHOD"  
INTERFACES

THAT'S EXACTLY  
WHERE  
LAMBDA FUNCTIONS  
COME IN

```
// Step 1: Create a list of strings
List<String> listOfStrings = new ArrayList<String>();

// Step 2: populate listOfStrings with names from a data file
// Won't bother with the code for this here

// Step 3: Sort this list of strings in lexicographical order,
// but with the added twist that if the name "Donald Trump" appears
// in this list, it must appear first.
Collections.sort(listOfStrings, new Comparator<String>() {
    public int compare(String s1, String s2) {
        if (s1.equals("Donald Trump") && !s2.equals("Donald Trump")) {
            return 1;
        } else if (!s1.equals("Donald Trump") && s2.equals("Donald Trump")) {
            return -1;
        }
        return s1.compareTo(s2);
    }
});
```

COMPARATOR IS AN INTERFACE  
WITH A SINGLE METHOD, COMPARE

IN OTHER WORDS, FOR THIS TYPE OF USAGE,  
SHORTHAND FOR A FUNCTION MIGHT BE MORE  
VALUABLE THAN SHORTHAND FOR A CLASS

NOW, INTERFACES WITH JUST ONE METHOD  
CAN ACTUALLY BE ENCAPSULATED USING  
SINGLE FUNCTIONS – NO REAL NEED FOR THEM  
TO BE ENCAPSULATED USING CLASSES

# LAMBDA FUNCTIONS ARE SIMPLY ANONYMOUS FUNCTIONS

# LAMBDA FUNCTIONS ARE SIMPLY ANONYMOUS FUNCTIONS

OBJECT ORIENTED LANGUAGES LIKE JAVA  
HAVE A WAY TO ENCAPSULATE OBJECTS  
INTO ANONYMOUS CLASSES

FUNCTIONAL LANGUAGES – LISP, PYTHON –  
HAVE A CORRESPONDING WAY TO  
ENCAPSULATE FUNCTIONS INTO  
ANONYMOUS FUNCTIONS CALLED  
LAMBDA FUNCTIONS OR LAMBDA  
EXPRESSIONS



IT IS WORTH MENTIONING HERE THAT THE  
ADVENT OF CLOUD COMPUTING HAS HELPED  
BRING FUNCTIONAL PROGRAMMING CONCEPTS  
BACK INTO VOGUE

IN A FOR-LOOP IT IS IMPOSSIBLE – OR AT LEAST  
VERY COMPLICATED – TO PARALLELIZE THE LOOP  
ACROSS MULTIPLE DIFFERENT CPUS

LAMBDA FUNCTIONS AND FUNCTIONAL  
PROGRAMMING ARE A NATURAL WAY TO  
PARALLELIZE COMPUTING ACROSS CPUS

THIS IS EXACTLY WHAT CLOUD  
COMPUTING IS ALL ABOUT

(DIVVY UP THE LIST YOU ARE ITERATING OVER,  
AND SEND PARTS OF THE LIST, ALONG WITH  
THE LAMBDA FUNCTION, TO DIFFERENT CLOUD  
NODES, THEN AGGREGATE THE RESULTS)