# Java + Concurrency

- Object oriented like C++ BUT
  - higher level of abstraction above machine
  - pros:
    - portability
    - reliability
    - concurrency
  - cons:
    - performance
- has byte codes (for abstract stack machines)
- interpreter + JIT compiler (in runtime)
  - for x86-64, ARM, etc.
- Single inheritance (simplicity + performance)

```
C++:
int* f() {
    int a[27];
    return a;
}

Java:
class C[extends D];
int[] m() {
    int a[2]; (arr of size 2)
    return a;
}
```

- The compiler checks if `a` escapes to caller or outside world
- size of array can be dedicated at runtime in Java
  - ex: `int a[n + f(x)];`
  - size is fixed once allocated
- primitive types - `bool, int, float`
- reference types - `Class, Thread, String, etc.` -> stored as pointer

A subclass method can shadow a superclass'

- exceptions:
  - abstract class/method -> declares itself but has no implementation
  - ex:

```
abstract class List {
    int l;
    abstract void append(Object o);
    int length() {return l;}
}

class LinkedList extends List {
    void append(Object o) {...}
}
```

  - we can't do `new List()` because there is no constructor, but we can do:
    - `List l = new LinkedList();`
  - as long as a class has at least one abstract method you can't create an instance of it

# Java Interfaces

```
interface X {
    int length();
    void append(Object o);
    ...
}

NO IMPLEMENTATIONS

interface Y extends X {
    int foo();
}
```

- a class can implement a single class but you can make as many interfaces as you want
  - bundling class + interface

final class (or method)

```
final int f(int a, int b) {
    return a * a + b;
```

```
}
```

- final class can't be subclassed
- final method can't be reimplemented
- pros:
    - efficiency? - replaces function calls with inline code for better instructions
    - trust

Object implementation:

```
class Object {
    public Object();
    public boolean equals(Object obj);
    public final Class getClass(); # runtime that represents the static notion of
class
    public int hashCode();
}
```