the ;

Java

```
class Cat
{
}
```

C++

```
class Cat
{
};
```

must end with ;

Java Generics vs C++ Templates

Java

```
class Stack<E>
```

C++

```
template <typename E>
class Stack
```

public, private, protected

Java

```
private int age;
private double gpa;
```

C++

```
private:                typical--create
    int age;            private section
    double gpa;
```

types

Java

```
char, int, float, double, boolean
```

C++

```
char, int, float, double, bool

(no Integer, Double, etc.)
```

main

Java

```
public static void main(String …)
```

must be in class

C++

```
int main()
```

not in class - in .cpp file

returns int - for us just 0

input/output

Java

```
System.out.println("..." + "...")
```

C++

```
#include <iostream>
using namespace std;

cout << "..." << "..." << endl;
```

const methods

Java

```
not available in Java
```

C++

```
int getGpa() const
{
    method cannot modify data mem
}
```

const & parameter passing

Java

```
not available in Java
```

C++

```
bool contains(const E& e)
{
    param e cannot be modified -
}   only call const methods of e
```

const combined

Java

```
not available in Java
```

C++

```
bool contains(const E& e) const
{
    only call const methods of e
}   do not modify this data mem
```

operator overloading

Java

cannot change meaning of ==, !=, +

```
boolean equals(Object other)
```

C++

```
bool operator==(const Cat& other)
```

== can now be used to compare Cats

operator overloading

Java

```
boolean equals(Object other) {
    if (this == other) { … }
    if (same type) { typecast }
```

C++

```
bool operator==(const Cat& other){
    if (this == &other) { … }
    else { no typecast }
```

pass by value (by copy)

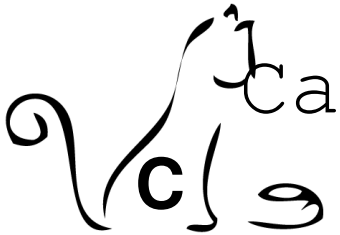
C++

```
bool contains(Cat x) {...}
```

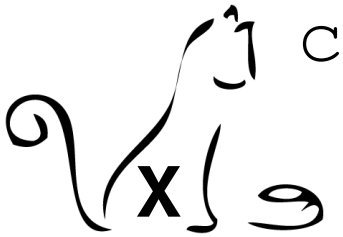
copy of `c` made for `x`

two Cats exist while `contains` runs

```
Cat c;
```




```
contains(c);
```



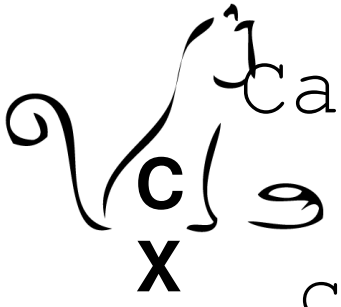
pass by reference & (no copy)

C++

```
bool contains(Cat& x) {...}
```

no copy – `x` attaches to `c`

one Cat exists while `contains` runs




```
Cat c;

contains(c);
```

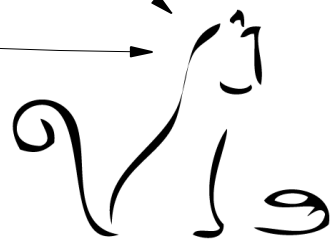
Java only has pass by value

Java

```
bool contains(Cat x) {...}


Cat c = new Cat();

contains(c);
```



- 1 Cat but 2 references/pointers
- the references are copied (passed by value) not the Cat