Java

class Cat
{
}

C++

class Cat
{
};  // must end with ;
.java vs .h, .cpp files

Java

```java
class Cat { rest of class }
```

C++

```cpp
#ifndef CAT_H
#define CAT_H
class Cat { rest of class };
#endif
```
Java Generics vs C++ Templates

Java

```java
class Stack<E>
```

C++

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

private int age;
private double gpa;

C++

private: int age;
private: double gpa;
public, private, protected

Java

private int age;
private double gpa;

C++

private:
int age;
double gpa;
Java reference vs C++ pointer

Java

Node node = new Node(...);
node.left = ...;

C++

Node* node = new Node(...);
node->left = ...;
Java null vs C++ nullptr

Java

    if (node == null)

C++

    if (node == nullptr)
types

Java
char, int, float, double, boolean

C++
char, int, float, double, bool
(no Integer, Double, etc.)
Java imports vs C++ includes

Java

```java
import java.util.Exception;
import java.util.Comparator;
```

C++

```cpp
#include <exception>

no Comparator – use <, >, ==
```
exceptions

Java

    throw new Exception();

C++

    throw exception();
exceptions

Java

```java
try {
}
catch (Exception e) { }
```

C++

```cpp
try {
}
catch (exception& e) { }
```
Java

```java
public static void main(String ...)
  must be in class
```

C++

```cpp
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++

```cpp
int getGpa() const
{
    // method cannot modify data mem
}
```
const & parameter passing

Java

not available in Java

C++

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

not available in Java

C++

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

Java

class Cat extends Pet

C++

class Cat : public Pet

multiple inheritance ok in C++
Java clone vs C++ copy c-tor

Java

```java
public Object clone() {
    build a copy of this Cat
}
need try/catch, Cloneable
```

C++

```cpp
Cat(const Cat& orig) {
    build this Cat out of orig
}
no try/catch, Cloneable
```
operator overloading

Java

cannot change meaning of ==, !=, +

boolean equals(Object other)

C++

bool operator==(const Cat& other)

== can now be used to compare Cats
Java

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

C++

```cpp
bool operator==(const Cat& other) {
    if (this == &other) { ... }
    else { no typecast }
}
```
garbage collection

Java

objects deleted/freed automatically

C++

must delete/free objects manually
delete myCat;
create destructor (constr. with ~)
~DLinkedList() { clear(); } 
destructor called automatically
pass by value (by copy)

C++

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

Cat c;
contains(c);
```

copy of c made for x

two Cats while contains runs
pass by reference (no copy)

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

Cat c;
contains(c);
```

x attaches to c – no copy
one Cat while contains runs
pass by value only in Java

Java

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

Cat c = new Cat();
contains(c);
```

1 Cat – 2 references/pointers
references copied (passed by value)