C# without the tough stuff

C#	Java
Pointers	NO Explicit Pointers
Delete	Garbage Collection
UD Type Conv	NONE
Op OV	Nope
Virtual - Multi Inherit.	Gone
Copy Constructor	Need is Gone (you can have them)
No →	•
No STRUCTS	

Similarities

C++ | Java

statements ; ✓

// /* */ ✓

if ( ) ✓

{ ✓

} ✓

while, do while, for ✓

Data types (fundamental)
int, float, char ✓

Functions ✓

NO: Unsigned

NO: Pass by Reference (&)

NO: Prototypes

Similarities & Differences

C#

Java

New - allocation ✓

Constructors ✓

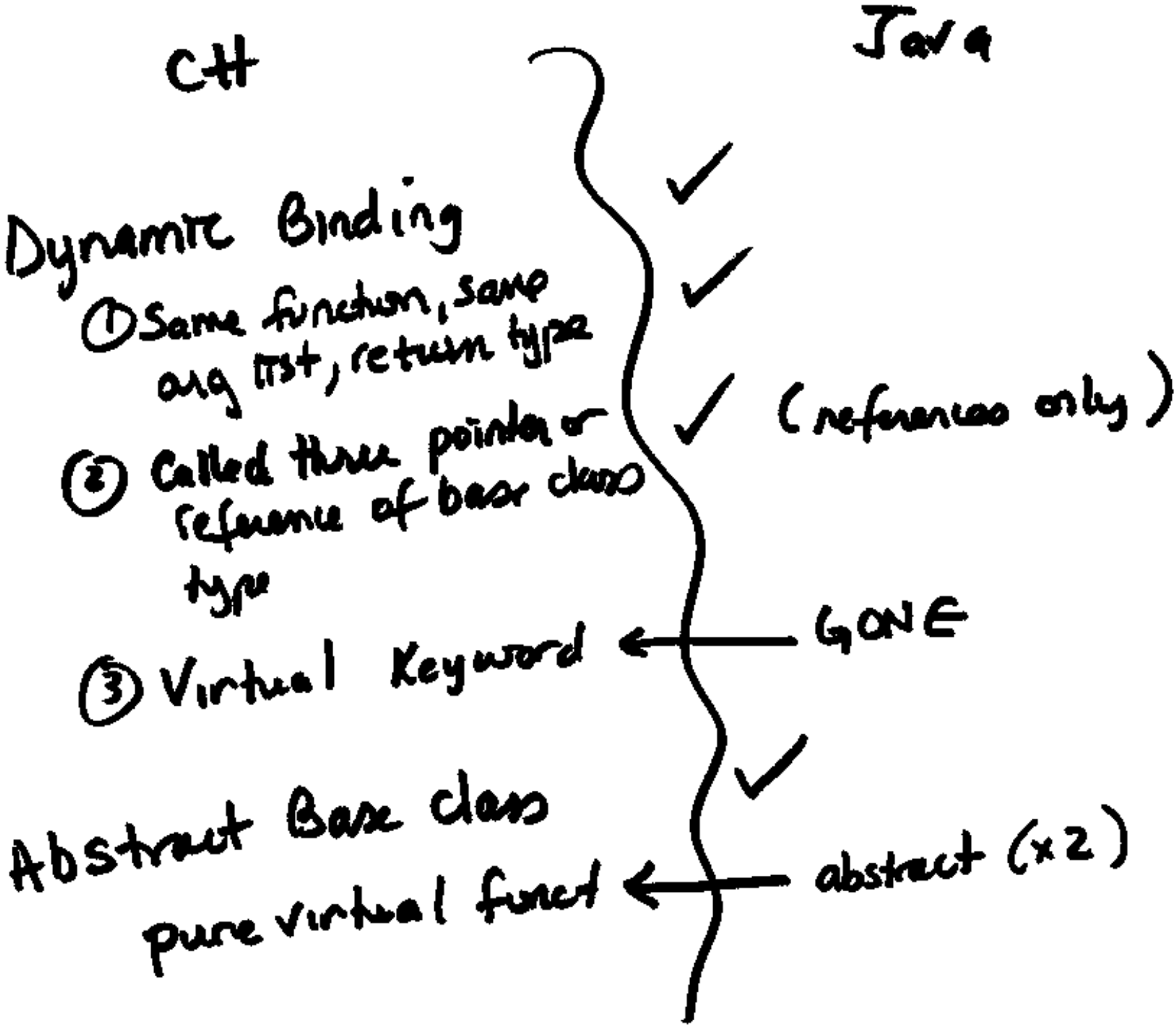
Function overloading ✓

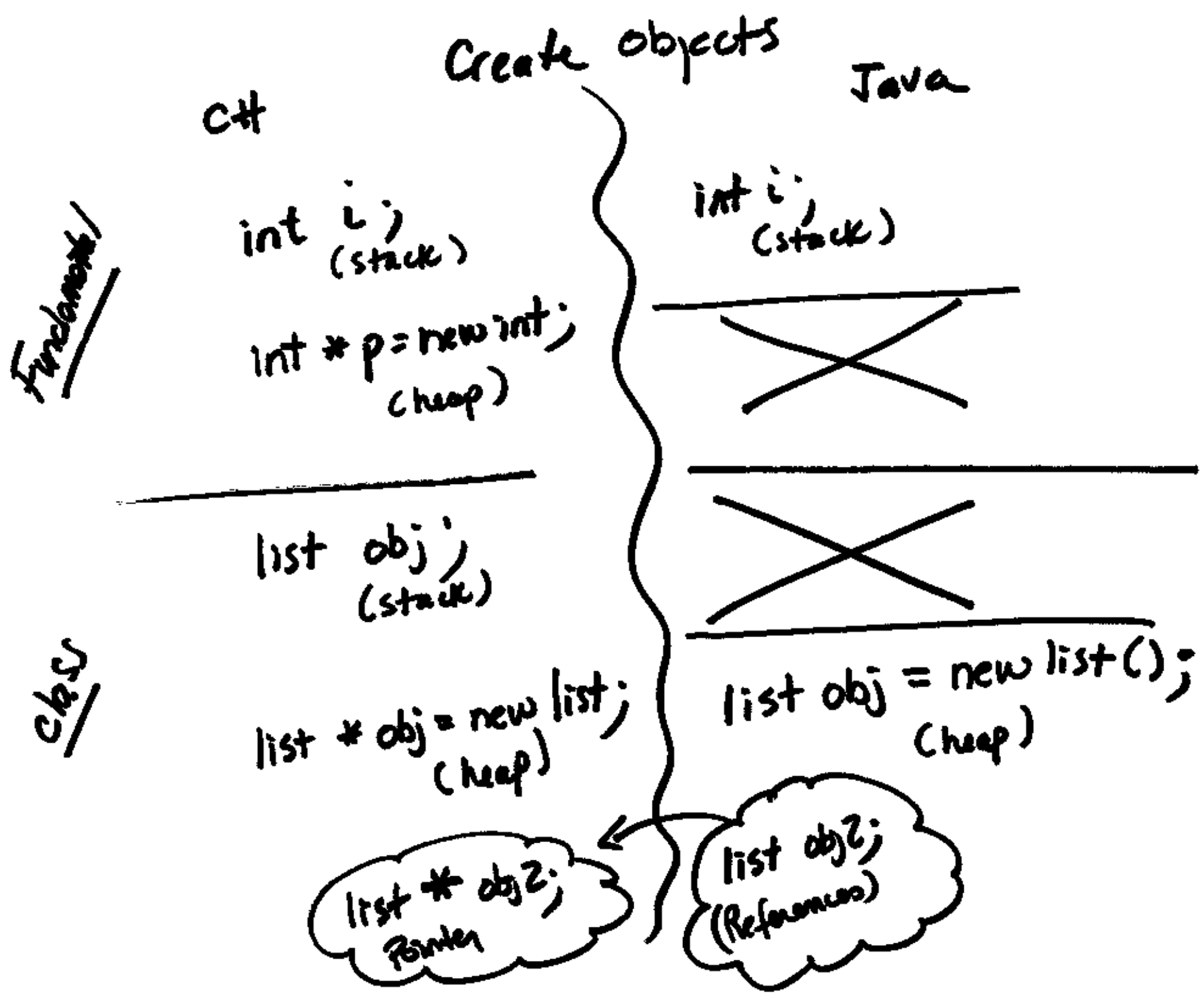
Class ✓ (Everything is in a class)

Inheritance ✓

Class derived : public base
{
};
↑ global variable

class derived extends base
{
} ← NO SEMICOLON


C++
Java
Dynamic Binding
① Same function, same arg list, return type
② Called thru pointer or reference of base class type
(references only)
③ Virtual Keyword
GONE
Abstract Base class
pure virtual funct
abstract (x2)

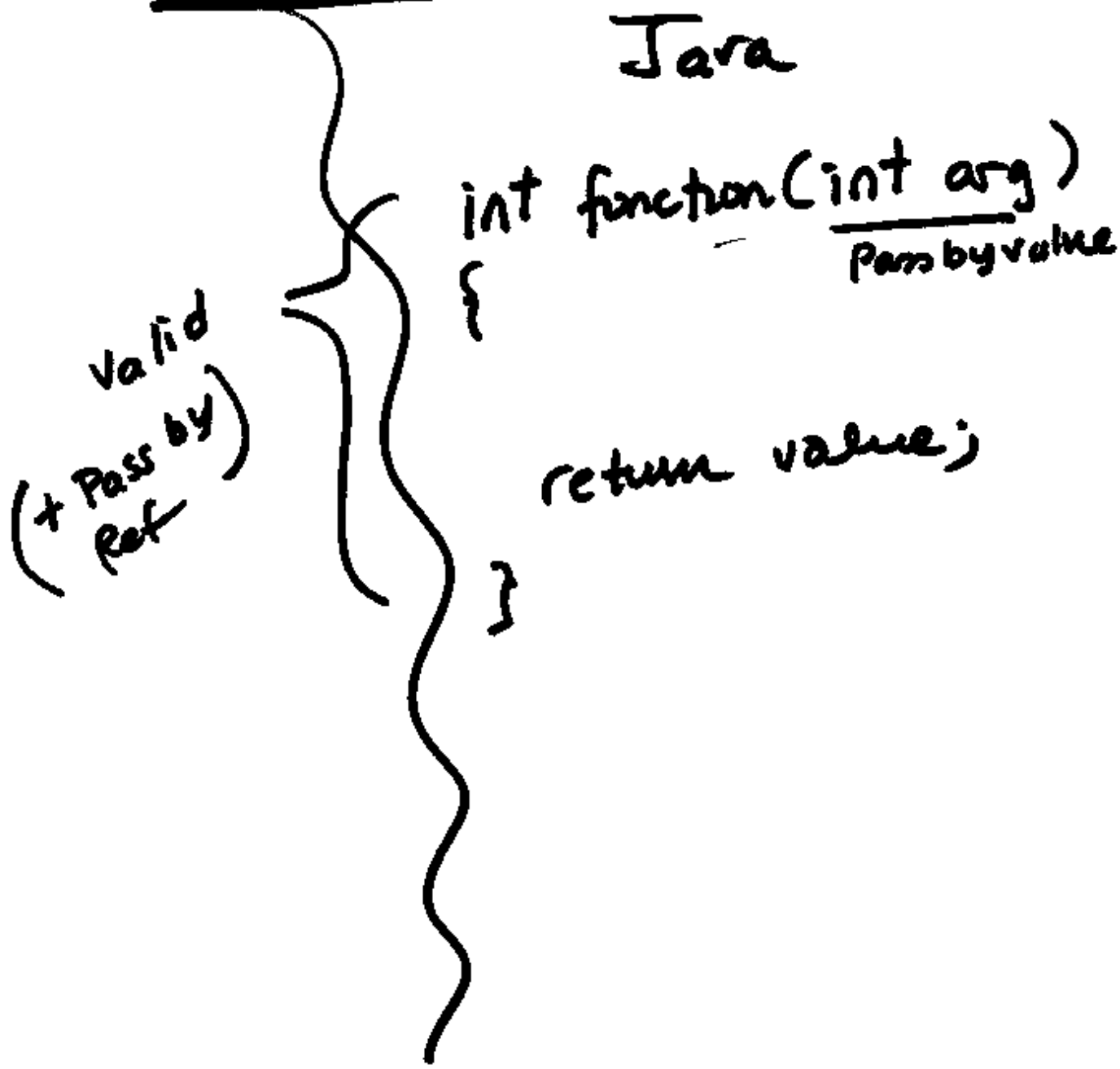

Create objects
C++
Java
Fundamental
int i;
(stack)
int i;
(stack)
int *p = new int;
(heap)
list obj;
(stack)
class
list *obj = new list;
(heap)
list obj = new list();
(heap)
list *obj2;
Pointer
list obj2;
(Reference)

C++
list **obj;
obj = new list;
int * array;
array = new int[num];
list ** array;
array = new list*[Size];
for(int i=0; i<Size; ++i)
array[i] = new list;
Java
(reference)
list obj;
obj = new list();
int array[];
// int [] array;
array = new int[Num];
list [] array;
array = new list[Size];
for(int i=0; i<Size; ++i)
array[i] = new list();
Pointer
Pointer
Pointer

Function - Fundamental

C++

Java



Functors - Class Types

C++

```
list * function(list *arg)
{
    ...
}
```

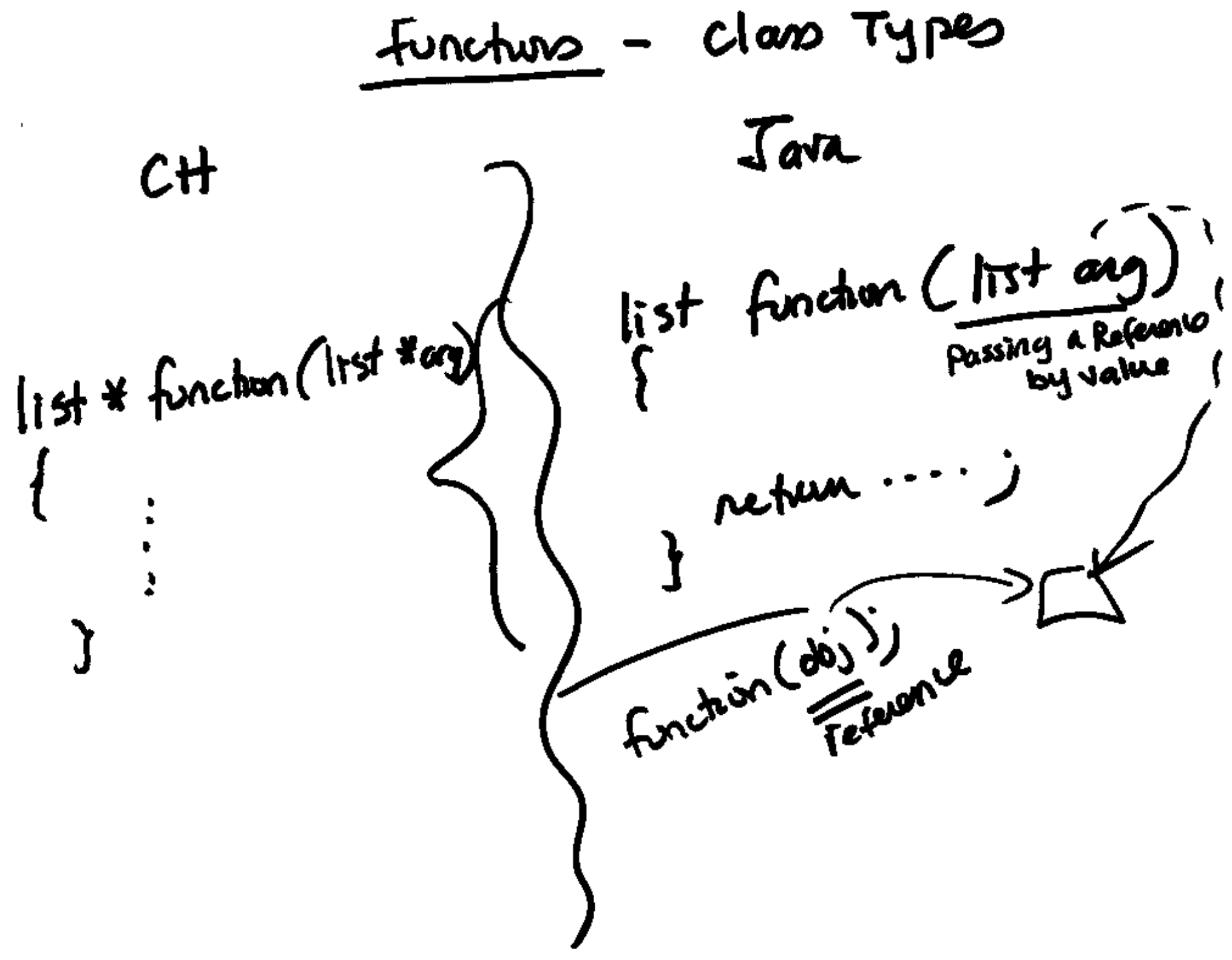
Java

```
list function(list arg)
{
    ...
    return ... ;
}
```

Passing a Reference by value

function(obj);

Reference



Recursion

C++

```
void insert(node*& root)
{   if (!root)
    {  root = new node;
          ⋮
    }
    else insert(root->left)
          ⋮
```

Java

```
node insert(node root)
{
  if(         )
  {
     root = new node();
  }
     return root;
  }
  else
     root[redacted]left = insert(root[redacted]left);
```

C++

```
class list
{  public:

   protected:

   privat:

};
```

Java


```
class list
{  private node root;

   public list() { ... body ... }
```