

Introduction to Java - Topic #8

"C# without the tough stuff!"

NO Operator Overloading

NO User Defined Type Conversions

NO explicit pointers

NO "delete"

NO destructors

NO "unsigned"

NO "Pass by Reference" (although we pass references by value)

} Garbage Collection!

Overview of Similarities C++ & Java

- 1) Built-in data types
(int, long, float, double, char
except **NO** unsigned)
- 2) Way objects are defined:
int i; ← Same
- 3) Loops: do while, while, for
(Same)
- 4) Operators: All arithmetic, relational,
logical, subscript, increment, decrement
except **NO** sizeof, explicit pointer ops,
scope resolution op.
- 5) Comments // and /* */
- 6) compound Blocks { }

Overview of Differences

- 1) **EVERYTHING** is in a class
- 2) **NO** Globals
- 3) **NO** Separate implementation versus prototype/declarations.
- 4) Methods are implemented **INSIDE** the class
- 5) **NO** friends

members are considered "friendly" if they are not preceded with private, protected, public keywords

these are **NOT** categories

- 6) **NO** Operator Overloading - Think I/O!
- 7) \emptyset is **NOT** FALSE - **while (head)**
— NO MORE —

Data Types

Primitive

int, long, short, etc.

ALLOCATED ON
The STACK

CANNOT be allocated
dynamically with new

Functions:

Can ONLY pass
and return by value

Reference

class types
arrays

ALLOCATED FROM
The HEAP

MUST be allocated
dynamically with new

(CAN'T be allocated on
the stack)

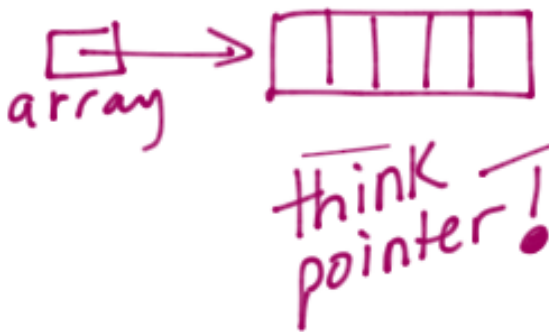
CAN ONLY pass and
return References
by Value

Arrays

C#

int array[5];

int * array;
array = new int[5];



Java

can't do this!

int array[];
array = new int[5];

NO size

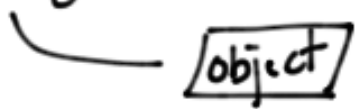
or
int [] array;
array = new int[5];



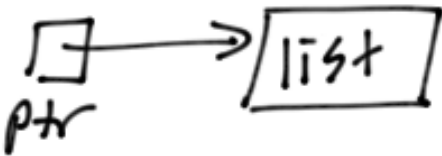
Class Types

C++

```
list object;
```




```
list * ptr;  
ptr = new list;
```



Java

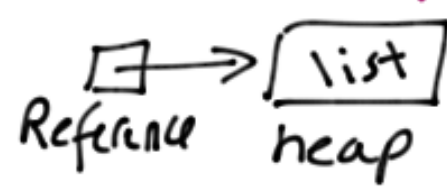
can't do this

```
list object;
```

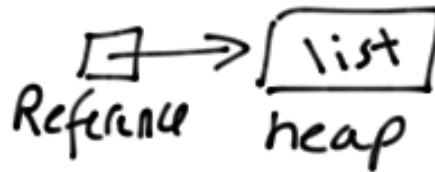


Creates a reference

```
object = new list();
```



need the parens!

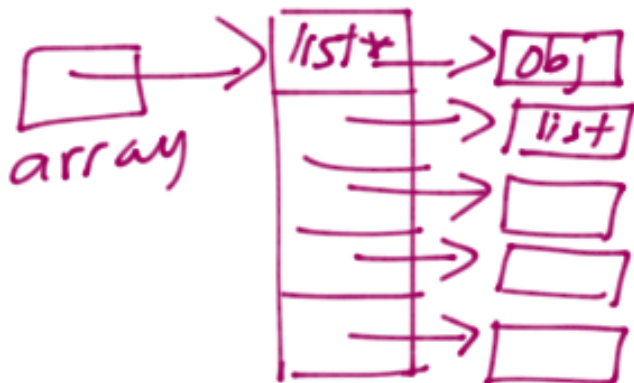


Array of class Types

C#

```
list array [5];
```

```
list ** array;  
array = new list* [5];  
for (int i = 0; i < 5; ++i)  
{  
    array[i] = new list;  
}
```



Java

```
can't do this!
```

```
list array [5];  
array = new list [5];
```

OR

an array of references
NOT objects

```
for (int i = 0; i < 5; ++i)  
{  
    array[i] = new list();  
}
```

new we
have list
objects!