Today - Lecture 7 - CS202

1) Discuss Java Topics
   - Review a few fundamental concepts
   - Functions & Arguments
   - Hierarchies
   - Dynamic Binding
   - Abstract Base classes
   - Interfaces
2) OO Design & Samples
3) Talk about the term paper

Next time
   a) Templates (C++)
   b) Prepare for Final Exam

Review some important Concepts:

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

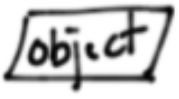
(CAN'T be allocated on the stack)

CAN ONLY pass and return References by Value

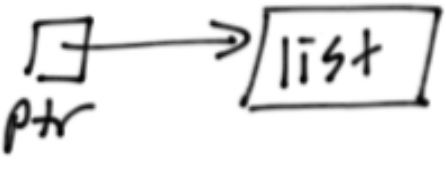
Class Types

C++

```
list object;
```




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



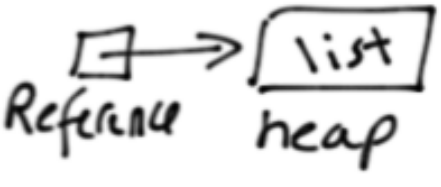
Java

Can't do this

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

creates a reference

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[];
```

OR

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

an array of references NOT objects

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

new we have list objects!

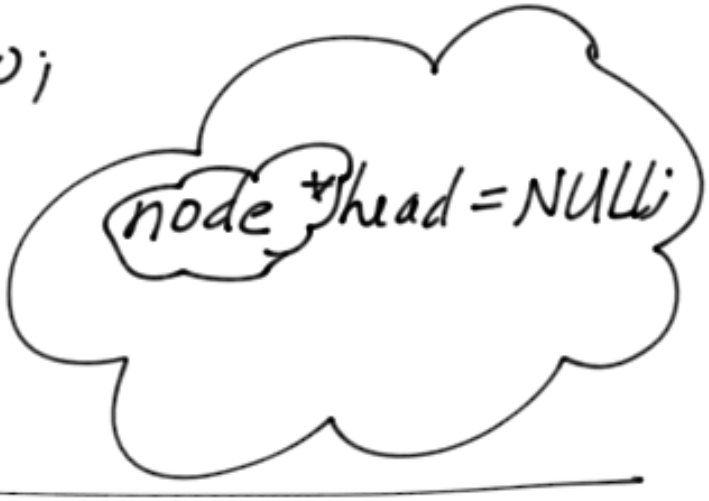
Everything is part of a class

```
class task
{   private String description;
    private int priority = 10;   <-- notice
    public task()
        {
            // implement body of
            // constructor here
        }
    <-- Add more [methods]
}   <-- notice
```

C++

```
void f()
{   int i = 10;
    if(        )
    {   int i = 100;
```

i = ?



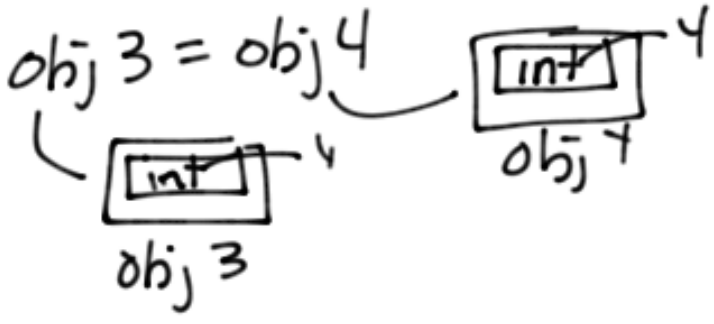
Java

```
public void f()
{
    ...
    this.head
```

Shallow vs Deep Copies

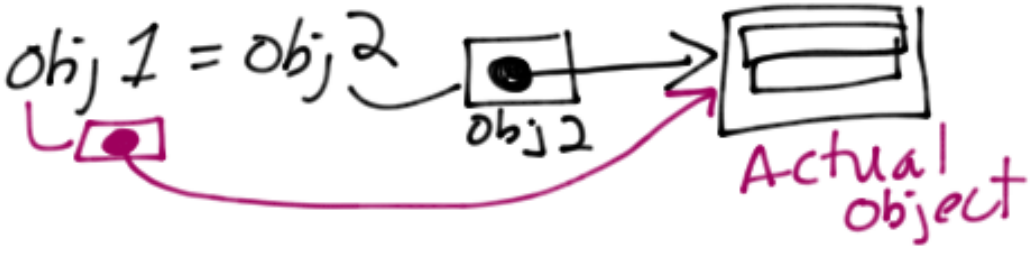
C#

"Shallow with a problem



"Shallow or membewise copy"

Java



C#

```
ptr1 = ptr2;
```

Functions (Methods)

C#

```
list function ( int a1, int &a2,
                list b1, list &b2,
                int array1 [],
                list objarray [] )
{
}
```

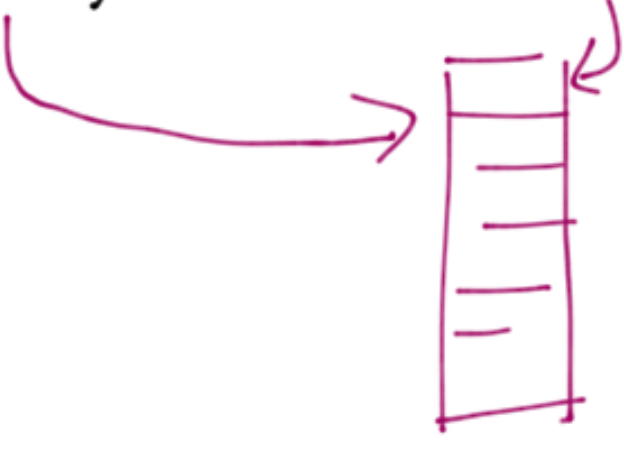
Java

```
public list function(int a1, list b2)
{
    int [] array1, int array2[],
}
```

copy

Pass by value

Passing a reference by value



```
//call
object = function(i, obj2);
```

Copy a BST

C++

```
void copy(node*& dest,
          node * source)
{
   if(source)
   {   dest = new node;
       // save data
       copy(dest->left, source->left);
       //
   }
   else  dest = NULL;
}
```

```
node * copy(node *dest, node* source)
{                    }

                dest->left = copy(dest->left, source->left);
                }
                return dest;
```

Copy a BST

Java

```
void copy(node dest,
          node source)
{
    if(source != null)
    {   dest = new node();
        // save data
        copy(dest.left, source.left);
        //
    }
    else  dest = NULL;
}
```

```
node copy(node source)
{
    if(source != null)
    {   node root = new node();
        // save the data
        root.left = copy(source.left)
                    getLeft()
        setLeft
        }
        return root;
```

Derivation

C#

```
class todo : public task
{

};
```

can be a comma separated list for multiple inheritance

Java

```
class todo extends task
{

}
```

can only be derived from one class and it is acts like public derivation