Lecture 3 Practice Problems

Practice – 1

1. List the 6 attributes of a variable.

2. Given the following Java code:

```
int __num__ = 10;
double num__$$ = 20.5;
int If = 30;
```

(a) Are the names of variables valid?

(b) Assume names are valid (or we modify to make them valid), for each of the variables please list its (known) attributes.

3. Assume a Java Demo class properly defined,

```
Demo d1 = new Demo("Atlanta"); // what is the type of d1?
Demo d2 = d1; //what is the relationship of d1 and d2?
```

Practice – 2

How many variables (including anonymous ones) are defined/created by the following code?

```
String s1;
String s2;
s2 = new String ("hello");
s1 = s2;
```

Practice – 3

- Based on lifetime of variables we classify variables into four categories. Please name these four categories.
- Given the following C++ code, for each of the variables please identify which category it belongs to.

```
void incrementAndPrint() {
        static int counter = 0; // Initialized only once
        counter++;
        std::cout << "Counter: " << counter << std::endl;
}

int main() {
        for (int i = 0; i < 5; ++i) { incrementAndPrint(); }
        return 0;
}
```

C++: support for allocation/deallocation

```
int main() {
int length;
cin >> length;
int * arr;
arr= new int[length];        //lifetime of arr?
for (int i = 0; i < length; i++) {
arr[i] = (i + 1) * 10;}

for (int i = 0; i < length; i++) {
cout << arr[i] << " " << endl;
}
delete[] arr;
return 0;
}
```

Java vs. C++: block scope

```
//C++
{
   int x = 4;
   {
     double x = 4.5;
      cout << x;
   }
   cout << x;
}
```

Q: what will be printed out if any?

```
//Java
{
   int x = 4;
   {
     double x = 4.5;
      System.out.println(x);
   }
   System.out.println(x);
}
```

Q: what will be printed out if any?

Global and Local Scope

```
//C++
int x = 10;
void func() {
     x = x + 10;      //local x or global x?
     printf("Value of x is %d\n", x);
}
int main() {
    func();
    x += 30;
    printf("Value of x is %d", x);      //value of x?
    return 0;
}
```

Scope – to be exact

```
//C++
double totalCost (double  items[], int count ) {                    //line1
        double margin = 20;                                         //line2
        double total = 0;                                           //line3
        for (int ct = 0; ct < count; ct++) {                        //line4
                double temp = items[ct] + margin;                   //line5
                total += temp;                                      //line6
        }                                                           //line7
        double taxRate = 0.095;                                     //line8
        total = total * (1+taxRate);                                //line9
        return total;                                               //line10
}                                                                   //line11
```

Q: Identify the scope of each variable defined above?

Scope – Java

```
int j;
for (j=0; j<5; j++) {
    System.out.println(j);
}
System.out.println(j);

//What will be printed out?
```

```
for (int j=0; j<5; j++) {
    System.out.println(j);
}
System.out.println(j);

//What will be printed out?
```

Nested Scope

```
#Python
def outer_function():
        x = 10
        def inner_function():
                x = 20
                print("x from inner function:", x)
        inner_function()
        print("x from outer function:", x)

outer_function()
```

Nested Scope -- Python

```
#Python
def outer_function():
        x = 10
        def inner_function():
                x = x + 10
                print("x from inner function:", x)
        inner_function()
        print("x from outer function:", x)

outer_function()
```

Nested Scope – Python's nonlocal

```
#Python
def outer_function():
        x = 10
        def inner_function():
                nonlocal x
                x = x + 10
                print("x from inner function:", x)
        inner_function()
        print("x from outer function:", x)

outer_function()
```

Nested Scope – Python's global

```
x=50
def outer_function():
    x = 10
    def middle_function():
        def inner_function() :
            global x
            x += 5
            print("x from inner function: ", x)
        inner_function()
        print("x from middle function:", x)
    middle_function()
    print("x from outer function", x)
outer_function()
print("global x", x)
```

Scope Resolution

```
x=50
def outer_function():
    x = 10
    def middle_function():
        def inner_function() :
            x += 5
            print("x from inner function: ", x)
        inner_function()
        print("x from middle function:", x)
    middle_function()
    print("x from outer function", x)
outer_function()
print("global x", x)
```

How can inner function access the x defined in the outer_function()?

Example: Static vs. Dynamic Scoping

```
function big() {
    var x;
    function sub1() {
        var x = 7;
        sub2();    //sub1 calls sub2
                   //assume call before function definition allowed
    } //end sub1

    function sub2() {
        print(x);          //uses x
    } //end sub2()

    x = 3;
    sub1();        //big calls sub1
} //end big
```

big calls sub1
sub1 calls sub2
sub2 uses x

- Static scoping
  - Reference to x in sub2 is to big's x
- Dynamic scoping
  - Reference to x in sub2 is to sub1's x