Joshua Catoe

CSCI 350

CSCI 350 Reflection Assignment on Memory Management Lesson

1. What are the 6 attributes of a variable?

**The six attributes of a variable are:**

**Name – human-readable identifier**

**Type – type of data variable will contain (int, double, char, etc.) (Also includes range of valid data values for types such as int.)**

**Value – actual value that variable will hold (5, 8.21, “J”, etc.)**

**Address – memory address of variable**

**Scope – visibility of variable throughout program**

**Lifetime – how long memory will be allocated for the variable**

1. What might be the pros and cons of designing the language to be case sensitive?  
     
   **The pros of a case-sensitive language, are fewer mistakes from the compiler and fewer mistakes from the programmer. The compiler will make fewer mistakes, as it will not have to make decisions such as “Is** name **the same variable as** Name**?”. The programmer will make fewer mistakes, as they will have to pay closer attention to what they are typing. The downside of this is that it would make the programmers job more time-consuming, having to constantly decide between upper and lower-case words.**
2. Explain the difference between static binding of variable type and dynamic binding of variable type.

**The key differences between static and dynamic variable type binding are that static occurs at compile time and allows for error checking and increased safety at the cost of performance and flexibility, while dynamic occurs at runtime and offers greater performance and flexibility at the cost of error checking and decreased safety.**

1. What do you think about the issue of flexibility vs safety/reliability in language design? Discuss the idea of protecting the programmer from his/her own mistakes. If you were designing a language for a specific purpose, how would that purpose influence your arguments above? What if you were designing a new all-purpose language?

**On the issue of flexibility vs safety/reliability, I believe it all depends on the purpose of the language in question. If I was designing a language that would power medical equipment in a hospital, I would want as few mistakes as feasibly possible and as many errors caught as feasibly possible, as people’s lives could be at stake for even the smallest errors. If I was designing a more general-purpose language, I would want to give the programmer as much freedom as possible, while also providing convenient error checking features. I admire what C# does, that is, have many safety features to protect the programmer, but also give them the freedom to use more powerful but riskier tools like pointers, if they so choose.**

1. For each variable declared and initialized below (x, y, b), list whether the variable is stored on the stack or heap?

static void Main(string[] args) {

static int x = 10;

int y = 5;

BankAccount b = new BankAccount();

}

**x – stored on Heap (Static vars. stored on heap.)**

**y – stored on Stack (Standard vars. stored on stack.)**

**b – stored on Heap (Objects stored on heap.)**

1. Draw the stack and the heap for the following program. When a stack frame is freed (when function execution ends), just draw a light “X” through the frame. Also, document the what the output would look like in the console window. (Do this without running the code first, then check it by running the code).

static int multiplier = 3;

static unsafe void Main(string[] args)

{

int g = 1;

int h = 1;

Console.WriteLine("g is {0}", g);

Console.WriteLine("h is {0}", h);

g = inc(g);

inc(h);

Console.WriteLine("g is {0}", g);

Console.WriteLine("h is {0}", h);

}

static int inc(int x){

int temp = x + 1;

int temp2 = inc2(x);

Console.WriteLine("temp is {0}", temp);

Console.WriteLine("temp with multiplier is {0}", temp \* multiplier);

return temp2;

}

static int inc2(int x){

int h = x + 2;

Console.WriteLine("h is {0}", h);

Console.WriteLine("h with multiplier is {0}", h \* multiplier);

return h;

}

}

inc2

Stack

h

x

3

g is 1

h is 1

h is 3

h with multiplier is 9

temp is 2

temp with multiplier is 6

h is 3

h with multiplier is 9

temp is 2

temp with multiplier is 6

g is 3

h is 1

1

inc

3

multiplier

3

temp2

temp

x

2

1

inc2

h

x

3

Heap

1

inc

temp2

temp

x

3

2

1

Console

main

h

g

1

3

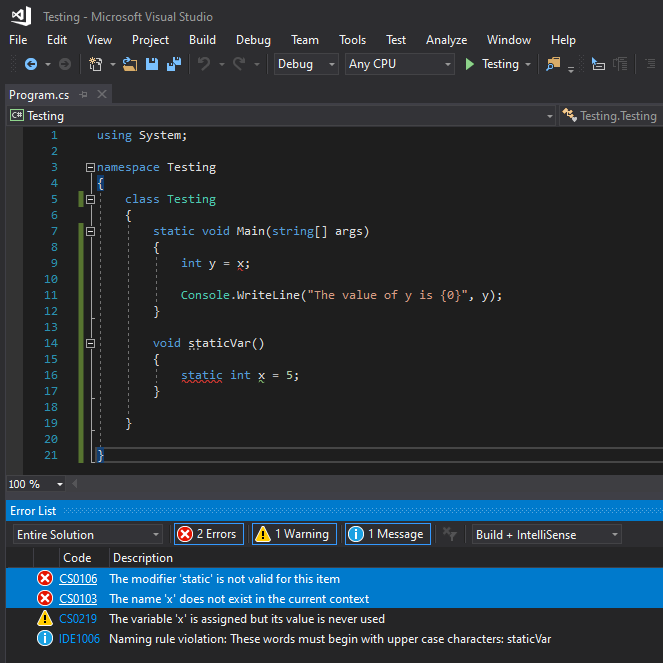
1. In the above code, what is the referencing environment of the statement “Console.WriteLine("g is {0}", g)”?

**The referencing environment of this statement is the variables** g **and** h**. Even though** h **is not used, it is still included in the referencing environment, as it can still be accessed.**

1. In the above code, what is the referencing environment of the statement   
   “int h = x + 2;”?

**The referencing environment of this statement is the variable** x **(the one passed to the function) and the variable** h **(the** h **declared in the function).**

1. Write a program (just a few lines) to demonstrate the scope of a static variable declared within a function. Is it visible outside of that function(such as in main) or is its scope local to the function? Show your findings and your code here.



**As can be seen in the screenshot, the second error states that** x **does not exist, thus it is not visible to** main**. The first error states that** static **is not valid in this statement; upon further research, it would seem that C# does not support local static variables.**

1. Write a program containing the following unsafe code. You’ll need to build it with the unsafe option allowed. Include a screenshot of the console with the output.

static unsafe void Main(string[] args)

{

int ab = 32;

int\* q = &ab;

int\* z = q;

Console.WriteLine("value of ab is {0}", \*q);

Console.ReadLine();

int var = 20;

int\* p = &var;

Console.WriteLine("Data is: {0} ", var);

Console.WriteLine("Data is: {0} ", p->ToString());

Console.WriteLine("Address is: {0} ", (int)p);

Console.WriteLine("<Your name>”);

}

