# Homework 3

Yuji Shimojo

CMSC 330

Instructor: Prof. Reginald Y. Haseltine

June 23, 2013

HOMEWORK 3

## Question 1

Scope and lifetime are distinct yet related issues in programming languages. Languages can sometimes make design decisions that cause a conflict between the scope and the lifetime of variables. Java's decision to allow classes to be defined inside a method illustrates this conflict. Consider the following example:

```
class AnonymousInnerClassInMethod
{

   public static void main(String[] args)
   {
      int local = 1;

      Comparable compare = new Comparable ()
         {
         public int compareTo(Object value)
            {
               return (Integer)value - local;
            }
         };
      System.out.println(compare.compareTo(5));
   }
}
```

Why does this code fail to compile? What could it be modified so it would compile? Explain the reason behind this failure in terms of scope and lifetime.

HOMEWORK 3

## My Answer

I tried to run the given sample above, and then I got a compile error. The reason of the error was that you cannot access to the local variable declared in main method inside the compareTo method. To fix it, two solutions are considered.

First is that you declare the local as a member variable of AnonymousInnerClassInMethod class as follows. In this case, the local variable must be static.

***Modified Code Snippet 1***

```
class AnonymousInnerClassInMethod
{
        static int local = 1;

   public static void main(String[] args)
   {...}
}
```

Second is that you add final modifier to the local as follows. In this case, you cannot change value of it while running this program.

***Modified Code Snippet 2***

```
public static void main(String[] args)
{
   final int local = 1;
```

```
    Comparable compare = new Comparable ()
        {
        public int compareTo(Object value)
            {...}
        };
    System.out.println(compare.compareTo(5));
}
```

## Question 2

C++ allows pointers to stack-dynamic variables. Consider the following C++ function:

```
int twice(int x)
{
    int *y;
    *y = x * 2;
    return *y;
}
```

Will the above function compile in C++? Is it correct? If not, under what circumstances will it fail and how should it be corrected? Consider one other language that has pointers. Does that language have the same problem? Explain.

## My Answer

First, I tried to run the program using the method as follows.

HOMEWORK 3

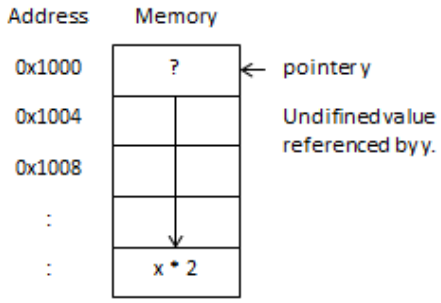
```
#include <iostream>

int twice(int x)
{
   int *y;
   *y = x * 2;
   return *y;
}

int main (int argc, const char * argv[])
{
   return 0;
}
```

I could compile it, but I had an EXC_BAD_ACCESS warning at the line of *y = x * 2. That means to suggest you try to access an already freed object as shown by Figure 1 below.

***Figure 1***



HOMEWORK 3

I modified to fix the program as the following Modified Program and Figure 2.

***Modified Program***

```
int twice(int x)
{
   int *y; // Declares a pointer
   int num1; // Declares an int variable
   y = &num1; // Assigns address of num1 to pointer y
   *y = x * 2; // num1 = x * 2
   printf("num1=%d\n",num1); // Shows value of num1
   printf("*y=%d\n",*y); // Shows value pointed by y, that is, value of num1
   printf("y=%p\n",y); // Shows value of y, that is, address of num1
   return *y;
}

int main (int argc, const char * argv[])
{
   int num2; // Declares an int variable num2
   num2 = twice(1); // num2 = 1 * 2
   printf("num2=%d\n", num2);
   return 0;
}
```

***Output***

```
num1=2
*y=2
```

HOMEWORK 3

y=0x7fff5fbff35c

num2=2

***Figure 2***