Homework 4

Yuji Shimojo

CMSC 330

Instructor: Prof. Reginald Y. Haseltine

July 7, 2013

Question 1

What is the output of the following Java program? Explain in terms of how parameters are passed in Java.

```
import java.awt.Point;

class PointParameters
{
   public static void main(String [] args)
   {
      int x = 1, y = 1;
      Point p = new Point(x, y), q = new Point(x, y);
      doubleScale(x, y, p, q);
      System.out.println( "(x,y) = " + new Point(x, y) +
         " p = " + p + " q = " + q);
   }

   private static void doubleScale(int x, int y, Point p, Point q)
   {
      x *= 2;
      y *= 2;
      p.x *= 2;
      p.y *= 2;
      q = new Point(x, y);
   }
}
```

Answer 1

Output

```
(x,y) = java.awt.Point[x=1,y=1] p = java.awt.Point[x=2,y=2] q = java.awt.Point[x=1,y=1]
```

Considerations

Before doubleScale() method is executed, memory allocation is as shown in Figure 1. Variable x, y, reference of p, and reference of q are copied to the arguments of doubleScale() in main method. In doubleScale() method, different local variables x and y, and a Point object q are created as shown in Figure 2.

Figure 1

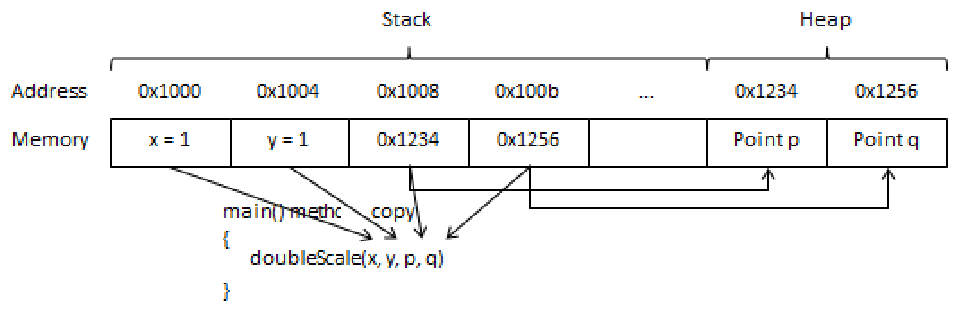
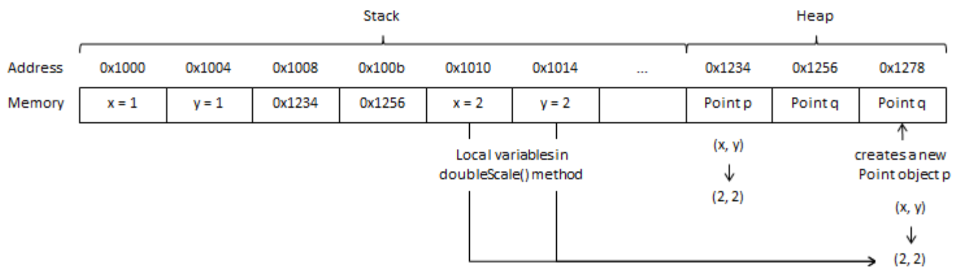


Figure 2



Question 2

Suppose a similar program was written in C# in which all the parameters were ref parameters. What would the output of that program be?

Answer 2

C# Program

```
using System;
namespace CMSC330.HW4
{
    class Point
    {
        public int x, y;
        public Point(int x, int y)
        {
            this.x = x;
            this.y = y;
        }
        public override string ToString()
        {
            return "(" + x + ", " + y + ")";
        }
    }
    class PointParameters
    {
        static void Main(string[] args)
        {
            int x = 1, y = 1;
            Point p = new Point(x, y), q = new Point(x, y);
            doubleScale(ref x, ref y, ref p, ref q);
            System.Console.WriteLine( "(x,y) = " + new Point(x, y) +
                " p = " + p + " q = " + q);
        }
        static void doubleScale(ref int x, ref int y, ref Point p, ref Point q)
        {
            x *= 2;
            y *= 2;
            p.x *= 2;
            p.y *= 2;
            q = new Point(x, y);
        }
    }
}
```

Output

```
(x,y) = (x=2, y=2) p = (x=2, y=2) q = (x=2, y=2)
```

Question 3

Examine the following C++ program, in which a IntList class is defined that contains an overloaded [] operator. What is the output of this program?

```
#include <iostream>
using namespace std;

class IntList
{
private:
   int list[1];
public:
   IntList() {list[0] = 0;}
   int& operator[] (const int index) {return list[index];}
};

int main()
{
   IntList list;

   cout << list[0] << endl;
   list[0] = 1;
```

```
    cout << list[0] << endl;
    return 0;
}
```

Output

```
0
1
```

Question 4

Notice that the overloaded [] operator returns a reference. Change the [] operator to return a value instead and explain what happens in that case. Explain why the ability to return by reference is essential to be able to implement such an operator. Can anything similar be done in Java?

Answer 4

After changing the [] operator to return a value, you will get a compile error at the line of list[0] = 1 in main method. This is because list[0] equals 1, so it’s not allowed to assign 1 to 1. In Java, basically primitives are passed by value and objects are passed by reference, so such a problem rarely happens.