Exam Prep 2: Another Deep Look at Our PPM Code

Melvyn Ian Drag

October 29, 2019

Abstract

In this class we'll discuss Pass-by-value vs. Pass-by-reference as well as take another look at how to set and unset individual bits in an *int*. We will do these things as we look at our PPM manipulator code again.

1 Garbage Collection, Primitives and References

1.1 Primitives

Draw a picture showing how these are real ones and zeros in memory. When you pass a primitive to a function, the ones and zeros are copied to a new location. So, whatever you do to those ones and zeros only affects the copy. *Draw a picture illustrating this and have the students draw it too.*

1.2 Objects + References

Whenever you do something like

```
MyClass mc = new MyClass();
```

Java stores some 1s and zeros in memory to fit all the data needed by the my class object. It does one more thing - Java creates a reference. It stores an additional 4 bytes in memory that contain the location of that object in memory. The way those 4 bytes work is outside of the scope of this class - you need to understand pointers. Then, when you pass your object to a function, Java makes a copy of the thing being passed to the function...

But the thing that is copied is not the memory that corresponds to the MyClass object - the thing that is copied is the *reference* to that thing. So, it you make a modification to the reference, you are actually modifying the object globally. *Draw a picture and allow for a brief discussion.*

1.3 Garbage Collection

A major feature of Java comes from the fact that Java manages object references - you get **Garbage Collection**! *Draw a picture of the RAM filling with stuff as more and more objects are "newed up". why doesn't the computer crash? It doesn't crash because Java deletes the objects after using them so they don't fill your RAM. You may have sene your computer get really slow if you have too many browser tabs open - that's because there's no RAM left over and the computer gets 'jammed up'.* This is one of the main features of the Java language - other languages may not clean up the old 1s and 0s for you. Of course, the people who write languages without Garbage collection call the Java programmers babies who can't keep track of the memory usage of their programs. The Java programmers might fire back that the only reason that garbage collection was created is because the people who wrote programs with out GC wrote such sloppy code that Garbage collection became very imporant. No matter your opinion, Garbage collection is one of the defining features of Java that is made possible by the fact that Java manages object references and not raw memory (except with primitives).

The fact that Java manages references to objects and not raw objects leads us to another defining feature of Java:

2 Pass By Value

From the *javaworld.com* reference below:

> Java does manipulate objects by reference, and all object variables are references. However, Java doesn't pass method arguments by reference; it passes them by value.

What does that mean?!?

When you pass a parameter to a function in Java, the JVM copies the parameter into the function - it does not pass the actual object in.

The learner is encouraged to study Java for another 6 months, then go learn about pointers in C++ and C to fully appreciate the difference between pass by value and pass by reference.

It is difficult to see that Java is pass by value, because all Java objects are references. The concept is simple, but difficult to explain without the learner understanding the way a few other programming languages work. Instead of explaining, I'll ask you to do the following assignment and see what strange things happen.

2.1 How this relates to the PPM manipulating code from last week

Look back at the function from the Week8 Code:

```java
  public static void ChangeWhiteToRed(int[][] inputArray){
    // ...
    // Note the void return type.
    // Yet the inputArray[][] is still modified after this function
    // has been called. That may surprise you - you might have
       expected
    // that the inputArray would not be modified, and that you would
    // have to return an int[][] from the function.
    // ...
  }
```

I want you to think more about this strange function. And so you can complete the following extra credit assignment.

2.2 Extra Credit #3

Create a Java program with four functions:

1. *public static void f1(float f)*
2. *public static void f2(float[] fArr)*
3. *public static void f3(CustomClass c)*
4. *public static void f4(CustomClass2 c)*

And write two classes

```java
public class CustomClass{
  public int x;
  CustomClass(int x){
    this.x = x;
  }
}
```

and

```java
public class CustomClass2{
  public CustomClass cc;

  CustomClass2(CustomClass cc){
    this.cc = cc;
  }
}
```

The functions *f1* - *f4* should change the paramters passed to them. More specifically,

1. *f1* should change the value of *f*.
2. *f2* should change the values in *fArr*.
3. *f3* should change the value of *c.x*.
4. *f4* should change the value of *c.cc.x*.

Check the parameter values before and after the function values ask Ask yourself:

Which values were changed in the functions? Which ones did not change?

You should see that the only value that is not changed in the function is the float f.

2.3 Reference

See this reference for more information: `https://www.javaworld.com/article/2077424/learn-java-does-java-pass-by-reference-or-pass-by-value.html`

3 Setting Bits In an Int

The essence of the midterm exam is to set a single bit in every int in an *int[][]*. As we mentioned, in our steganography project we need to set the last bit in every int. More generally, something you might want to do in your code is set a bit, any one of the 32 bits in an int. How do you do that?

To set the last bit you do this:

```java
public int setLastBit(int i){
  return ( i | 0x00000001 );
}
```

to turn off the last bit in an int you would:

```java
public int zeroLastBit(int i){
  return ( i & 0xFFFFFFFE );
}
```

The above functions would be very useful to your midterm assignment. . .

3.1 Extra Credit #4a

Write general implementations of the above. Implement the following function

```java
/**
 * Set a specified bit to either 1 or 0
 *
 * @param originalValue - integer you want to modify
 * @param bitIndex - the bit ( 0 - 31 ) you want to change
 * @param value - New bit value. true for 1, false for 0.
 */
public int setBit(int originalValue, int bitIndex, boolean value){
  // Implement me.
  // somehow create a newValue with the proper bit set to the
     proper val.
  return newValue;
}
```

The answer is here: `https://stackoverflow.com/questions/4674006/set-specific-bit-in-byte`

3.2 Extra Credit #4b

As the above is too simple - I've given a link with code you can copy and paste - you must do a little more work to earn these points. Copy the *tricky* and *main* functions from here: `https://www.javaworld.com/article/2077424/learn-java-does-java-pass-by-reference-or-pass-by-v` `html`. Compile, run and explain the output to me.

4 Also might be helpful for Midterm

If, while doing the Midterm assignment, you need to extract the nth bit from a byte, you can do this:

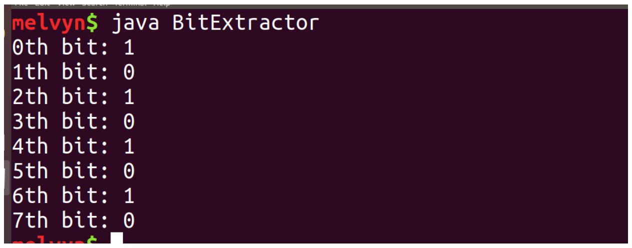
```java
public static byte getBit(byte b, int index)
{
   return (byte)((b >> index) & 0x01);
}
```

You can verify that this works by checking the output for byte 0b01010101 = 0x55.

```java
public class BitExtractor{
  public static byte getBit(byte b, int index)
  {
     return (byte)((b >> index) & 0x01);
  }

  public static void main(String[] args){
    byte b = (byte)0x55;
    for( int idx = 0; idx < 8; ++idx){
      System.out.println(String.format("%dth bit: %d", idx,
         getBit(b, idx)));
    }
  }
}
```

Output:



```
melvyn$ java BitExtractor
0th bit: 1
1th bit: 0
2th bit: 1
3th bit: 0
4th bit: 1
5th bit: 0
6th bit: 1
7th bit: 0
```

Figure 1: Showing the bits 0-7 being extracted from 0x55

5 Midterm

The midterm assignment is to:

1. Read an ASCII PPM image into an int[][]

2. Grab the bits one by one from an ASCII string

3. One by one, set the last bit in the (red?) channel of the PPM image using the bits from the step above.
4. Write the image out to a file.
5. Read the image back into memory.
6. Verify that the data was properly stored in the last bits of the red channel.

Last week I gave a function that read a PPM into an int[][].
Last week we saw a function that writes the int[][] out to a PPM file.
Above you have functions for extracting bits and setting bits.
Good luck!

6 Coming up next

Over the next two weeks we'll look at some Java Collections.

7 Final Exam Rough Draft

Put on github. In the next few weeks we'll drill everything together to make sure you're ready.