Week # 4 – Exercises / Solutions

Exercise #1: Consider the following class definition:

```
public class Complex {
     private double re;
     private double im;
     private static int instanceCount = 0;

     public Complex(double real, double imag)
          re = real;
          im = imag;
     }
}
```

- Identify the Local Variables in the Complex constructor?
- Identify the Instance Variables in the above Class definition?
- Identify the Class Variables in the above Class definition?
- For all the local variables identified in the above Complex constructor, indicate their slot numbers. Also, discuss if a local variable needs more than one slot and why?

Local Variables

- **this** reference (as non-static)
- **real** and **imag** (parameters)

Instance Variables

- **re** and **im**

Class Variables

- **instanceCount** as declared static

Slot Numbers

- **this** reference – Slot # 0
- **real** – Slot # 1 (needs to two slots, as its double; actually takes slots 1 & 2)
- **imag** – Slot # 3 (needs to two slots, as its double; actually takes slots 3 & 4)

Exercise #2: Show how the following values would be stored in a byte-addressable machines with 32-bit words, using little endian and then big endian format. Assume that each value starts at address 1016. Draw a diagram of memory for each, placing the appropriate values in the correct (and labeled) memory locations.

- 0x456789A1
- 0x0000058A
- 0x14148888

For 0x456789A1

Memory Address	**1016**	**1017**	**1018**	**1019**
Little Endian	A1	89	67	45
Big Endian	45	67	89	A1

For 0x0000058A

Memory Address	**1016**	**1017**	**1018**	**1019**
Little Endian	8A	05	00	00
Big Endian	00	00	05	8A

For 0x14148888

Memory Address	**1016**	**1017**	**1018**	**1019**
Little Endian	88	88	14	14
Big Endian	14	14	88	88

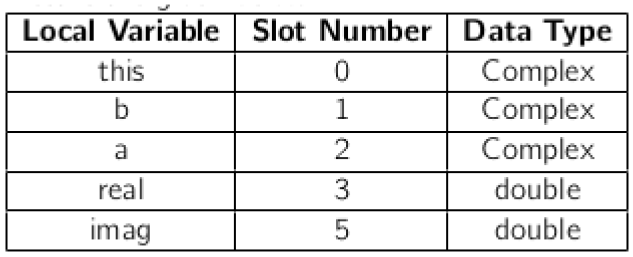
Exercise #3: Now, introduce the following method definitions to the Complex class seen above, and then use javap to disassemble the compiled bytecode (.class) file.

```
public Complex plus(Complex b){
     Complex a = this;
     double real = a.re + b.re;
     double imag = a.im + b.im;
     return new Complex(real, imag);
}

public static void main(String[] args){
     Complex a = new Complex(5.0, 6.0);
     Complex b = new Complex(-3.0, 4.0);
     System.out.println(”a + b = ” + a.plus(b));
}
```

- How many local variables are created in the stack frame of plus method? List all of them and mention their slot numbers as well as data types.

A total of five local variables are created on the stack frame of plus method.



Local Variable	Slot Number	Data Type
this	0	Complex
b	1	Complex
a	2	Complex
real	3	double
imag	5	double

- Which instructions in the bytecode for **plus method** correspond to the Java source code?

  The plus method from above class is compiled into bytecode as follows:

```
public Complex plus(Complex);
  Code:
     0: aload_0
     1: astore_2
     2: aload_2
     3: getfield       #2                  // Field re:D
     6: aload_1
     7: getfield       #2                  // Field re:D
    10: dadd
    11: dstore_3
    12: aload_2
    13: getfield       #3                  // Field im:D
    16: aload_1
    17: getfield       #3                  // Field im:D
    20: dadd
    21: dstore         5
    23: new            #4                  // class Complex
    26: dup
    27: dload_3
    28: dload          5
    30: invokespecial  #5                  // Method "<init>":(DD)V
    33: areturn
```

  The bytecode to Java source code mapping is given below:

Bytecode Lines	**Java Source Code**
0-1	a = this;
2-11	real = a.re + b.re;
12-21	imag = a.im + b.im;
23-33	return new Complex(real, imag);

- What is the purpose of invokespecial instruction in the bytecode of plus method?

  The **invokespecial** instruction at the end of bytecode is required to create a new object of Complex type, which is then returned to the calling method.