

INFO-C307 Laboratory Session #2

LAB GOALS

- Understanding datatypes and bitwise operators

Step 1: Start with a simple Java program. Create a class called `Datatypes_Bitwise_Operators`

Step 2: Add a new (main) function with the following. Note that at this point all the functions are commented. Our goal for this lab is to better understand our data type and how we can represent and manipulate them.

```
public static void main(String[] args) {  
    //JavaDataTypes();  
    //DisplayingInDifferntNumberSystems();  
    //Binary_Representation_1();  
    //Binary_Representation_2();  
    //Bit_Manipulation();  
}
```

Step 3: The first thing we want to do is to understand our basic data types and how they are constructed in our compiler. Add the following function below the main() function, then uncomment the function call `JavaDataTypes();` in the main() function and run it. This function allows us to understand the built-in types provided by Java and how many bits is used for each type. In addition, we can see the range of numbers(data) that can be represented in each type.

```
//-----  
public static void JavaDataTypes() {  
    System.out.println("-----");  
    System.out.println("----- Java Data Types -----");  
    System.out.println("-----");  
  
    System.out.println("Type\t #bits\t Min / Max");  
    System.out.println("-----\t -----");  
    System.out.println(Byte.TYPE + "\t " + Byte.SIZE + "\t " + Byte.MIN_VALUE + " to " + Byte.MAX_VALUE);  
    System.out.println(Short.TYPE + "\t " + Short.SIZE + "\t " + Short.MIN_VALUE + " to " + Short.MAX_VALUE);  
    System.out.println(Character.TYPE + "\t " + Character.SIZE + "\t " + (int)Character.MIN_VALUE + " to " + (int)Character.MAX_VALUE);  
    System.out.println(Integer.TYPE + "\t " + Integer.SIZE + "\t " + Integer.MIN_VALUE + " to " + Integer.MAX_VALUE);  
    System.out.println(Long.TYPE + "\t " + Long.SIZE + "\t " + Long.MIN_VALUE + " to " + Long.MAX_VALUE);  
    System.out.println(Float.TYPE + "\t " + Float.SIZE + "\t " + Float.MIN_VALUE + " to " + Float.MAX_VALUE);  
    System.out.println(Double.TYPE + "\t " + Double.SIZE + "\t " + Double.MIN_VALUE + " to " + Double.MAX_VALUE);  
}
```

```
-----  
----- Java Data Types -----  
-----  
  
Type      #bits    Min / Max  
-----  
byte      8        -128 to 127  
short     16       -32768 to 32767  
char      16        0 to 65535  
int       32       -2147483648 to 2147483647  
long      64       -9223372036854775808 to 9223372036854775807  
float     32       1.4E-45 to 3.4028235E38  
double    64       4.9E-324 to 1.7976931348623157E308
```

Step 4: The next thing we should learn is to display information in different formats. This will come in quite handy as we get into lower-level programming. Sure you have a debugger which may be able to show you this data. But you should really know how to do this yourself, and without a debugger. Add the following function `DisplayingInDifferntNumberSystems()` to your program, and un-comment the corresponding function call in your `main()` function.

```
public static void DisplayingInDifferntNumberSystems() {  
  
    System.out.println("-----");  
    System.out.println("----- Representing Data Using Different Number Systems -----");  
    System.out.println("----- Char, Hex, Dec, Octal -----");  
    System.out.println("-----");  
    System.out.println(Byte.TYPE + "\t" + Byte.SIZE + "\t" + Byte.MIN_VALUE + " to " + Byte.MAX_VALUE);  
  
    byte b = 0x7f;  
  
    System.out.println("'b' As byte is " + b);  
  
    // printf function (inherited from C and C++)  
    System.out.printf("'b' As Character is %c\n", b);  
    System.out.printf("'b' As Hex is %x\n", (int) b);  
    System.out.printf("'b' As Integer is %d\n", (int) b);  
    System.out.printf("'b' As Octal is %o\n", (int) b);  
}
```

```
-----  
----- Representing Data Using Different Number Systems -----  
----- Char, Hex, Dec, Octal -----  
-----  
byte      8      -128 to 127  
'b' As byte is      127  
'b' As Character is  
'b' As Hex is      7f  
'b' As Integer is   127  
'b' As Octal is     177  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Notice the use of `printf()` function. `Printf()` is a function that comes from “C” language. It came across to C++ and now Java. `Printf()` is quite flexible as you will find out...

```
// printf function (inherited from C and C++)  
System.out.printf("'b' As Character is %c\n", b);  
System.out.printf("'b' As Hex is %x\n", (int) b);  
System.out.printf("'b' As Integer is %d\n", (int) b);  
System.out.printf("'b' As Octal is %o\n", (int) b);
```

Step 5: Now let's create a function `Binary_Representation_1()` to display the actual bit pattern of our data. Some data types, (int and long) have a `toBinaryString()` method that allows us to view the bit patterns in variables/values of this type. Also notice that other data types may not have the `toBinaryString()` functions available to them.

```
public static void Binary_Representation_1() {
    System.out.println("-----");
    System.out.println("-Built-in Integer to Binary Conversion-");
    System.out.println("-----");
    System.out.println("Int 10 to binary: " + Integer.toBinaryString(10));
    System.out.println("Long 10 to binary: " + Long.toBinaryString(10));

    // There is no .toBinaryString() for these other data types.
    // Uncomment the lines below (one at a time and see for yourself)
    //
    //System.out.println("Byte 10 to binary: " + Byte.toBinaryString(10));
    //System.out.println("Short 10 to binary: " + Short.toBinaryString(10));
    //System.out.println("float 10.0 to binary: " + float.toBinaryString(10.0f));
}
```

```
-----
-Built-in Integer to Binary Conversion-
-----

Int 10 to binary: 1010
Long 10 to binary: 1010
BUILD SUCCESSFUL (total time: 0 seconds)
.
```

Step 6: So, what about the rest of the data types? Can we create our own function that allows us to display the content of a data type? (after all its only memory!) Well, let try it. Let's create a function `Binary_Representation_2()` to display the actual bit pattern of our data. We will try this with Integers, characters and floats. (See below) Add the function `Binary_Representation_2()` to your program, and uncomment its call in your main().

```
public static void Binary_Representation_2() {
    System.out.println("-----");
    System.out.println("-User Defined Binary Conversion-");
    System.out.println("-----");

    System.out.println("Int 10 to binary:      " + int_to_binary(10,Integer.SIZE));

    System.out.println("Char A to binary:      " + char_to_binary('A', Character.SIZE));

    System.out.println("Float 3.5f to binary: " + float_to_binary(3.5f, Float.SIZE));
    System.out.println("Float -3.5f to binary: " + float_to_binary(-3.5f, Float.SIZE));
}
```

Note that you will get some errors if you try to compile the above. That is because the above function makes use of some “helper” functions to help it display the bit pattern. These functions are:

```
int_to_binary(10,Integer.SIZE)
char_to_binary('A',Character.SIZE)
float_to_binary(3.5f,Float.SIZE)
```

Let's add these functions to our program as well. Note that these functions use BIT manipulation operators to get at each bit within the variable and create a string with the equivalent bit pattern and return it to us.

(read and study the code in the next page and then add them to your program) (We will examine bitwise operation in the next step so, if you don't completely understand what is going on, don't worry!)

```

public static String int_to_binary(int value, int size) {
    String theResult = "";
    int mask = 0x80000000;
    int i;
    for (i = 0; i < size; i++) {
        if ((value & mask) == 0) {
            theResult = theResult.concat("0");
        } else {
            theResult = theResult.concat("1");
        }
        if((i+1)%8 == 0) theResult = theResult.concat(" ");
        //System.out.println(theResult);
        //System.out.println(mask);
        mask >>= 1;        // Shift right (zero fill)
    }
    return (theResult);
}

//-----
public static String char_to_binary(char value, int size) {
    String theResult = "";
    char mask = 0x8000;
    int i;
    for (i = 0; i < size; i++) {
        if ((value & mask) == 0) {
            theResult = theResult.concat("0");

        } else {
            theResult = theResult.concat("1");
        }
        if((i+1)%8 == 0) theResult = theResult.concat(" ");
        //System.out.println(theResult);
        //System.out.println(mask);
        mask >>= 1;        // Shift right (zero fill)
    }
    return (theResult);
}

//-----
public static String float_to_binary(float floatValue, int size) {
    String theResult = "";
    int mask = 0x80000000;
    int value = Float.floatToIntBits(floatValue);    // conver the float bits into an equivalent integer
    int i;
    for (i = 0; i < size; i++) {
        if ((value & mask) == 0) {
            theResult = theResult.concat("0");

        } else {
            theResult = theResult.concat("1");
        }
        if((i+1)%8 == 0) theResult = theResult.concat(" ");
        //System.out.println(theResult);
        //System.out.println(mask);
        mask >>= 1;        // Shift right (zero fill)
    }
    return (theResult);
}

```

Now that you have all the necessary components, run the program and you should see the following result for step 6.

```
-----  
-User Defined Binary Conversion-  
-----  
Int 10 to binary:      00000000 00000000 00000000 00001010  
Char A to binary:      00000000 01000001  
Float 3.5f to binary:  01000000 01100000 00000000 00000000  
Float -3.5f to binary: 11000000 01100000 00000000 00000000  
BUILD SUCCESSFUL (total time: 0 seconds)  
.
```

Step 7: Now let's perform some bit manipulation using bitwise operators (&, |, ^, ~, >>, <<, >>>). These operations allow us to do low level bit operations. Add the function **Bit_Manipulation()** to your program and uncomment its call in the main() function.

```
public static void Bit_Manipulation() {
    System.out.println("-----");
    System.out.println("----- Bit Manipulation -----");
    System.out.println("--- &, |, ^, ~, <<, >>, >>> ----");

    int i = 0x00000001;

    int theAND = i & 0x000000ff;
    int theOR = i | 0x000000ff;
    int theXOR = i ^ 0x000000ff;
    int theNOT = ~(0x000000ff);
    int theRightShift = 0x000000ff >> 1;
    int theLeftShift = 0x000000ff << 2;

    System.out.println("0x00000001 = " + (0x00000001));
    System.out.println("0x00000001 = " + int_to_binary( (0x00000001),Integer.SIZE));
    System.out.println("0x000000ff = " + (0x000000ff));
    System.out.println("0x00000001 = " + int_to_binary( (0x000000ff),Integer.SIZE));
    System.out.println("AND: 0x00000001 & 0x000000ff = " + (i & 0x000000ff));
    System.out.println("AND: 0x00000001 & 0x000000ff = " + int_to_binary( (i & 0x000000ff),Integer.SIZE));
    System.out.println("AND: 0x00000001 & 0x000000ff = " + int_to_binary( theAND,Integer.SIZE));
    System.out.println("OR: 0x00000001 | 0x000000ff = " + int_to_binary( theOR,Integer.SIZE));
    System.out.println("XOR: 0x00000001 ^ 0x000000ff = " + int_to_binary( theXOR,Integer.SIZE));
    System.out.println("COMPLEMENT ~0x00000001 = " + int_to_binary( theNOT,Integer.SIZE));
    System.out.println("SHIFT-RIGHT: 0x000000ff >> 1 = " + int_to_binary( theRightShift,Integer.SIZE));
    System.out.println("SHIFT-LEFT: 0x000000ff << 2 = " + int_to_binary( theLeftShift,Integer.SIZE));
}
```

```
-----
----- Bit Manipulation -----
--- &, |, ^, ~, <<, >>, >>> ----
0x00000001 = 1
0x00000001 = 00000000 00000000 00000000 00000001
0x000000ff = 255
0x00000001 = 00000000 00000000 00000000 11111111
AND: 0x00000001 & 0x000000ff = 1
AND: 0x00000001 & 0x000000ff = 00000000 00000000 00000000 00000001
AND: 0x00000001 & 0x000000ff = 00000000 00000000 00000000 00000001
OR: 0x00000001 | 0x000000ff = 00000000 00000000 00000000 11111111
XOR: 0x00000001 ^ 0x000000ff = 00000000 00000000 00000000 11111110
COMPLEMENT ~0x00000001 = 11111111 11111111 11111111 00000000
SHIFT-RIGHT: 0x000000ff >> 1 = 00000000 00000000 00000000 01111111
SHIFT-LEFT: 0x000000ff << 2 = 00000000 00000000 00000011 11111100
BUILD SUCCESSFUL (total time: 0 seconds)
```

What to Submit

- Source code
- Screen shot showing your results after every step
- A short paragraph indicating what new ideas you learned from this lab.
- Did you find any errors? Did you find any part of the lab confusing?

The entire source code for this lab can be found below:

```
package datatypes_bitwise_operators;

/**
 *
 * @author hhakimzadeh
 */
public class Datatypes_Bitwise_Operators {

    public static void main(String[] args) {
        JavaDataTypes();
        //DisplayingInDifferntNumberSystems();
        //Binary_Repesentation_1();
        //Binary_Repesentation_2();
        //Bit_Manipulation();
        //Displaying_Data_In_Different_Formats_1();
    }

    //-----
    public static void JavaDataTypes() {
        System.out.println("-----");
        System.out.println("----- Java Data Types -----");
        System.out.println("-----");

        System.out.println("Type\t #bits\t Min / Max");
        System.out.println("----\t ----\t -----");
        System.out.println(Byte.TYPE + "\t " + Byte.SIZE + "\t " + Byte.MIN_VALUE + " to " + Byte.MAX_VALUE);
        System.out.println(Short.TYPE + "\t " + Short.SIZE + "\t " + Short.MIN_VALUE + " to " + Short.MAX_VALUE);
        System.out.println(Character.TYPE + "\t " + Character.SIZE + "\t " + (int) Character.MIN_VALUE + " to " +
            (int) Character.MAX_VALUE);
        System.out.println(Integer.TYPE + "\t " + Integer.SIZE + "\t " + Integer.MIN_VALUE + " to " +
            Integer.MAX_VALUE);
        System.out.println(Long.TYPE + "\t " + Long.SIZE + "\t " + Long.MIN_VALUE + " to " + Long.MAX_VALUE);
        System.out.println(Float.TYPE + "\t " + Float.SIZE + "\t " + Float.MIN_VALUE + " to " + Float.MAX_VALUE);
        System.out.println(Double.TYPE + "\t " + Double.SIZE + "\t " + Double.MIN_VALUE + " to " + Double.MAX_VALUE);
    }

    //-----
    public static void DisplayingInDifferntNumberSystems() {

        System.out.println("-----");
        System.out.println("----- Representing Data Using Different Number Systems -----");
        System.out.println("----- Char, Hex, Dec, Octal -----");
        System.out.println("-----");
        System.out.println(Byte.TYPE + "\t " + Byte.SIZE + "\t " + Byte.MIN_VALUE + " to " +
            Byte.MAX_VALUE);

        byte b = 0x7f;

        System.out.println("'b' As byte is      " + b);

        // printf function (inherited from C and C++)
        System.out.printf("'b' As Character is %c\n", b);
        System.out.printf("'b' As Hex is      %x\n", (int) b);
        System.out.printf("'b' As Integer is   %d\n", (int) b);
        System.out.printf("'b' As Octal is     %o\n", (int) b);
    }
}
```

```

//-----
public static void Binary_Representation_1() {
    System.out.println("-----");
    System.out.println("-Built-in Integer to Binary Conversion-");
    System.out.println("-----");
    System.out.println("Int 10 to binary: " + Integer.toBinaryString(10));
    System.out.println("Long 10 to binary: " + Long.toBinaryString(10));

    // There is no .toBinaryString() for these other data types.
    // Uncomment the lines below (one at a time and see for yourself)
    //
    //System.out.println("Byte 10 to binary: " + Byte.toBinaryString(10));
    //System.out.println("Short 10 to binary: " + Short.toBinaryString(10));
    //System.out.println("float 10.0 to binary: " + float.toBinaryString(10.0f));
}

//-----
public static void Binary_Representation_2() {
    System.out.println("-----");
    System.out.println("-User Defined Binary Conversion-");
    System.out.println("-----");

    System.out.println("Int 10 to binary:      " + int_to_binary(10, Integer.SIZE));

    System.out.println("Char A to binary:      " + char_to_binary('A', Character.SIZE));

    System.out.println("Float 3.5f to binary:  " + float_to_binary(3.5f, Float.SIZE));
    System.out.println("Float -3.5f to binary: " + float_to_binary(-3.5f, Float.SIZE));
}

//-----

public static String int_to_binary(int value, int size) {
    String theResult = "";
    int mask = 0x80000000;
    int i;

    for (i = 0; i < size; i++) {
        if ((value & mask) == 0) {
            theResult = theResult.concat("0");
        } else {
            theResult = theResult.concat("1");
        }
        if ((i + 1) % 8 == 0) {
            theResult = theResult.concat(" ");
        }
        //System.out.println(theResult);
        //System.out.println(mask);
        mask >>= 1;          // Shift right (zero fill)
    }
    return (theResult);
}

```

```

//-----
public static String char_to_binary(char value, int size) {
    String theResult = "";
    char mask = 0x8000;
    int i;

    for (i = 0; i < size; i++) {
        if ((value & mask) == 0) {
            theResult = theResult.concat("0");

        } else {
            theResult = theResult.concat("1");
        }
        if ((i + 1) % 8 == 0) {
            theResult = theResult.concat(" ");
        }
        //System.out.println(theResult);
        //System.out.println(mask);
        mask >>= 1;          // Shift right (zero fill)
    }
    return (theResult);
}

//-----
public static String float_to_binary(float floatValue, int size) {
    String theResult = "";
    int mask = 0x80000000;
    int value = Float.floatToIntBits(floatValue);    // convert the float bits into an equivalent integer
    int i;
    for (i = 0; i < size; i++) {
        if ((value & mask) == 0) {
            theResult = theResult.concat("0");

        } else {
            theResult = theResult.concat("1");
        }
        if ((i + 1) % 8 == 0) {
            theResult = theResult.concat(" ");
        }
        //System.out.println(theResult);
        //System.out.println(mask);
        mask >>= 1;          // Shift right (zero fill)
    }
    return (theResult);
}

```

```
//-----

public static void Bit_Manipulation() {
    System.out.println("-----");
    System.out.println("----- Bit Manipulation -----");
    System.out.println("--- &, |, ^, ~, <<, >>, >>> ----");

    int i = 0x00000001;

    int theAND = i & 0x000000ff;
    int theOR = i | 0x000000ff;
    int theXOR = i ^ 0x000000ff;
    int theNOT = ~(0x000000ff);
    int theRightShift = 0x000000ff >> 1;
    int theLeftShift = 0x000000ff << 2;

    System.out.println("0x00000001 = " + (0x00000001));
    System.out.println("0x00000001 = " + int_to_binary((0x00000001), Integer.SIZE));
    System.out.println("0x000000ff = " + (0x000000ff));
    System.out.println("0x00000001 = " + int_to_binary((0x000000ff), Integer.SIZE));
    System.out.println("AND: 0x00000001 & 0x000000ff = " + (i & 0x000000ff));
    System.out.println("AND: 0x00000001 & 0x000000ff = " + int_to_binary((i & 0x000000ff), Integer.SIZE));
    System.out.println("AND: 0x00000001 & 0x000000ff = " + int_to_binary(theAND, Integer.SIZE));
    System.out.println("OR: 0x00000001 | 0x000000ff = " + int_to_binary(theOR, Integer.SIZE));
    System.out.println("XOR: 0x00000001 ^ 0x000000ff = " + int_to_binary(theXOR, Integer.SIZE));
    System.out.println("COMPLEMENT ~0x00000001 = " + int_to_binary(theNOT, Integer.SIZE));
    System.out.println("SHIFT-RIGHT: 0x000000ff >> 1 = " + int_to_binary(theRightShift, Integer.SIZE));
    System.out.println("SHIFT-LEFT: 0x000000ff << 2 = " + int_to_binary(theLeftShift, Integer.SIZE));
}

//-----

public static void Displaying_Data_In_Different_Formats_1() {
    System.out.println("-----");
    System.out.println("---- Char, HEX, Dec, Octal ----");
    System.out.println("-----");

    char c = 'a';
    System.out.println("'a' As Character is " + c); // a
    System.out.println("'a' As Short is " + (short) c); // 97

    // printf function (inherited from C and C++)
    // Read more about printf and formatting your output
    // https://www.cs.colostate.edu/~cs160/.Summer16/resources/Java_printf_method_quick_reference.pdf
    System.out.printf("'a' As Character is %c \n", c);
    System.out.printf("'a' As Hex is %x\n", (short) c);
    System.out.printf("'a' As Hex is %08x\n", (short) c); //
    System.out.printf("'a' As Integer is %d\n", (int) c);
    System.out.printf("'a' As Integer is %08d\n", (int) c);
    System.out.printf("'a' As Octal is %o\n", (short) c);
    System.out.printf("'a' As Octal is %09o\n", (short) c);
}

}
```