Recommended Reading:

Java [Primitive Data Types](http://java.sun.com/docs/books/tutorial/java/nutsandbolts/datatypes.html)

Java [Operator Precedence Table](http://java.sun.com/docs/books/tutorial/java/nutsandbolts/operators.html)

|  |  |
| --- | --- |
| Line 1. | public class Quiz3{ |
| Line 2. | public static void main (String [] args){ |
| Line 3. | int d=21; |
| Line 4. | byte j=10; |
| Line 5. | int c=20; |
| Line 6. |  |
| Line 7. | System.out.println (d%2+"3"+j); |
| Line 8. | while (c>0){ |
| Line 9. | j\*=2; |
| Line 10. | d=c-d/2\*3-j; |
| Line 11. | System.out.println(d%2+"3"+j); |
| Line 12. | if (c<3){ |
| Line 13. | c--; |
| Line 14. | }else{ |
| Line 15. | c-=3; |
| Line 16. | } |
| Line 17. | } |
| Line 18. | d=c-d/2\*3-j; |
| Line 19. | System.out.println(d%2+"3"+j); |
| Line 20. | } |
| Line 21. | } |

|  |  |
| --- | --- |
| Line 3. | int d=21; |

This line tells the computer to create a variable (storage in the memory) named “d” to hold an integer number and then assign the value 21 to variable d.

21 is within the range of values (-2,147,483,648 🡪 2,147,483,647) that can be held by any variable of type int. So, d is able to store 21, and value of d becomes 21.

|  |  |
| --- | --- |
| Line 4. | byte j=10; |

This line tells the computer to create a variable (storage in the memory) named “j” to hold a byte number and then assign the value 10 to the variable j.

10 is within the range of values (-128 🡪 +127) that can be held by any variable of type byte. So, j is able to store 10, and value of j becomes 10.

|  |  |
| --- | --- |
| Line 5. | int c=20; |

The explanation is similar to the one for line 3.

|  |  |
| --- | --- |
| Line 7. | System.out.println (d%2+"3"+j); |

We have to first evaluate the value of the expression d%2+"3"+j

and then print the value.

Lets evaluate it step by step showing how Java does it:

d % 2 + "3" + j

21 % 2 + "3" + 10 (putting values of the variables)

According to the Java precedence table, modulus or remainder operator, % has higher precedence over additional operator, + (like the BODMAS rule)

21 % 2 + "3" + 10

1 + "3" + 10 (if we divide 21 by 2, the remainder is 1)

Now we have two ‘+’ operators. Which one will work first? The answer is most of the operators work from left to right hand side direction. This is called ‘associativity’. So, the first ‘+’ operator will work first. Its operands are 1 and "3". When a ‘+’ operator has a String one of the operands, it concatenates (puts them together) and creates a new String.

1 + "3" + 10

"13" + 10 (1 and "3" got concatenated and become a new string "13")

"1310" (explanation is same as above)

We have found the value of the expression. Now, this is printed to the System’s output, monitor.

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

Value of the variable c is 20. The number 20 is greater than zero. The condition is true, we go inside the loop.

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j\*=2 is a short hand for j = j \* 2. So, our actual expression is j = j \* 2; What java will do is, it will evaluate the expression (j \* 2) on the right hand side of the assignment operator (=) and then assign the value from the right hand side to the variable at the left hand side (j in this case).

j is a byte and 2 is an int.

Lets evaluate it:

j (byte) \* 2 (int)

10 (byte) \* 2 (int) (putting the values of the variables)

10 (int ) \* 2 (int)

the value byte 10, was promoted to int 10. This means it was converted to a compatible data type (int) which can hold at least the same values that a byte can hold. As ‘int’ can hold more, it’s a promotion!

10 (int ) \* 2 (int)

20 (int)

Now, our original statement was

j=j\*2;

We got the value of the right hand side. Now the statement looks like:

j (byte) =20 (int)

numbers on the right hand side got promoted to int during temporary calculation. For storing it inside j, a variable of type byte, the right hand side has to be a byte.

j(byte)=20 (int)

j(byte)=20 (byte)

20(int) was converted to 20(byte).The reason you find the value to be same, is there was no data loss (loss of precision). So, now the value of j is 20.

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

c - d / 2 \* 3 - j

20(int) - 21(int) / 2(int) \* 3(int) - 20(byte) (putting values)

/ and \* got same precedence, but / comes first from left to right

20(int) - 10(int) \* 3(int) - 20(byte)

20(int) - 30(int) - 20(byte)

There are two ‘-‘, we use the left most one first.

-10(int) - 20(byte)

We have to ‘-‘ operator. The first one is unary. It applies to the number 10 only. The 2nd one is binary. It applies to both -10 and +20.

-10(int) - 20(byte)

-10(int) - 20(int) (temporary data type promotion)

-30(int)

Now our statement is:

d(int) = -30 (int)

Data types are compatible, hence, -30 gets stored inside variable, d.

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d % 2 + "3" + j

-30(int) % 2(int) + "3" (String) + 20 (byte)

% has the highest precedence

0(int) + "3" (String) + 20 (byte)

There are two ‘+’ with equal precedence. We use the left most one (associativity).

"0"(String) + "3"(String) + 20 (byte)

"03"(String) + 20(byte)

"03"(String) + "20"(String)

"0320"(String)

Now 0320 is shown at the output.

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

The value of c is 20. Twenty is not less than three. The condition is false. We jump out of if condition/statement block, after ‘}’ on Line 14 and see there is an else block.

|  |  |
| --- | --- |
| Line 14. | }else{ |

As there is no further ‘if’ under this else, we follow whatever is written inside this else block (line 14->16), blindly.

|  |  |
| --- | --- |
| Line 15. | c-=3; |

c-=3 is a short hand for c=c-3;

c=c-3

c=20-3

c=17

We did not bother about data types because everything was in int and we are very far from overflowing / underflowing int.

|  |  |
| --- | --- |
| Line 16. | } |

This is the closing for else block. We are done with else block.

|  |  |
| --- | --- |
| Line 17. | } |

We find that it is the closing for while loop. We cannot go out of while loop unless the condition is false. We go back to line 8 to check if the condition of while loop is false or not. If false, we will jump out of while loop block (line 8 🡪 line 17), if true, we will keep following the contents of the while loop.

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

c is 17, which is greater than 0. Condition is true, we go inside of the while loop.

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j\*=2

j=j\*2

j(byte) = j(byte) \* 2(int)

j(byte) = 20(byte) \* 2(int)

j(byte) = 20(int) \* 2(int)

j(byte) = 40(int)

j(byte) = 40(byte)

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d(int) = c (int) – d (int) / 2(int) \* 3(int) - j(byte)

d(int) = 17(int) – (-30) (int) / 2(int) \* 3(int) - 40(byte)

d(int) = 17(int) – (-15) (int) \* 3(int) - 40(byte)

d(int) = 17(int) – (-45) (int) - 40(byte)

d(int) = 62(int) - 40(byte)

d(int) = 62(int) - 40(int)

d(int) = 22(int)

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+"3"+j

22%2+"3"+40

0+"3"+40

"03"+40

"0340"

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

c is 17, condition is false, lets go to else block.

|  |  |
| --- | --- |
| Line 14. | }else{ |

inside else block, we find line 15.

|  |  |
| --- | --- |
| Line 15. | c-=3; |

c=c-3

c=17-3

c=14

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

condition true, going in…

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j\*=2

j=j\*2

j=40 (byte) \* 2 (int)

j=40 (int) \* 2 (int)

j (byte) = 80 (int)

j (byte) = 80 (byte)

80 is well within the range of data that a byte can hold.

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=14-22/2\*3-80

d=14-11\*3-80

d=14-33-80

d= -19-80

d= -99

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+"3"+j

-99%2+"3"+80

-1+"3"+80 (if you divide -99 by two, the remainder will be minus 1)

"-13"+80

"-1380"

This is our 4th output.

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

c is 14, condition false, we won’t follw if statement block. We will follow else block.

|  |  |
| --- | --- |
| Line 14. | }else{ |
| Line 15. | c-=3; |
| Line 16. | } |

c-=3

c=c-3

c=14-3

c=11

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

condition true, going in..

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j\*=2

j=j\*2

j(byte) = j(byte) \* 2(int)

j(byte) = 80(byte) \* 2(int)

j(byte) = 80(int) \* 2(int)

j(byte) = 160(int)

Now there is a problem. How do we convert 160 from int to byte? byte variables are not that big that it will be able to hold 160. It can’t hold 160. It can only hold -128 🡪 +127.

Simply, some part of the data will be lost. We have to find out what is finally getting stored back to j.

There are several ways to determine the value. You can experiment and find your own one as well.

Shammur made it easy for you and told you about few ways (read older emails from cse110 group)

The hardest/geekiest/nerdy/genius’s way to understand is to analyze it in binary numbering system.

I hope to discuss this in details. In case I forget, pls do remind me to do so. But we can wait till the midterm is over, to become a genius. The following is what I have found out, thanks to Shammur.

Byte can hold only 256 numbers (from -128 to +127)

byte b;

int x;

int y;

x=160; // x could be any value outside range of byte

the line "b=(byte) x;" will work like following:

y=x%256;

if(y>127) {

y=y-256;

}

if(y<-128) {

y=y+256;

}

b=(byte) y;

In simpler words, (remember only these, nothing else)

* you mod / calculate remainder by 256
* If this is <=127 OR >=-128, it is your answer,
* else,
  + you subtract 256 from it, if it is positive
  + or add 256 to it, if it is negative and get the answer.

Additional attached files show how I found these out. Most likely, you still have confusion. Please feel free to write back.

If you would prefer a pseudo code for the procedure. Here it is,

largeVariable = largeValue % TotalNo;

if(largeVariable > posMax) {

largeVariable = largeVariable - TotalNo;

}

if(largeVariable < negMin) {

largeVariable = largeVariable + TotalNo;

}

smallVariable=(smallType) largeVariable;

where,

TotalNo = number of numbers that our smaller data type can hold. For byte, it is 256

posMax = maximum positive number. For byte, it is +127

negMin = minimum negative number. For byte, it is -128

largeVariable = A variable which can hold values for temporary calculation, say of type int.

smallVariable=A variable, where we want to put a large value. Its b, for byte b=160.

largeValue=the value that we are trying to put inside the smaller variable, 160.

smallType=The type of our small variable. In this case, it is byte.

Hope I did not screw you yet. Where were we?

j(byte) = 160(int)

160 % 256

160 is greater than 127,

160-256

-96

j(byte) = -96(int)

j(byte) = -96(byte)

now j stores the value -96.

Though this is not what we intended to do. We intended to store 160. But byte can’t store 160. It stored what it could. If we really want to store 160, we should use short/int/long/char. Only these can store the value 160.

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=11-(-99)/2\*3-(-96)

d=11-(- 49)\*3-(-96)

d=11-(-147)-(-96)

d= 158-(-96)

d=254

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+”3”+j

254%2+”3”+(-96)

0+”3”+ (-96)

”03”+ (-96)

”03-96”

This is our 5th output.

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

c is 11, condition false, look for else part

|  |  |
| --- | --- |
| Line 14. | }else{ |
| Line 15. | c-=3; |
| Line 16. | } |

c=c-3

c=11-3

c=8

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

Who says 8 is not greater than 0?

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j=j\*2

j(byte) = j(byte) \* 2(int)

j(byte) = -96(byte) \* 2(int)

j(byte) = -192(int)

Can byte hold -192? The answer is NO.

-192 % 256

-192

-192 is less than 127

-192+256

64

j(byte) = +64 (int)

j(byte) = 64 (byte)

j becomes 64

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=8-254/2\*3-64

d=8-127\*3-64

d=8-381-64

d=-373-64

d=-437

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+”3”+j

-437%2+”3”+64

-1+”3”+64

”-13”+64

”-1364”

Isn’t this our 6th output?

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

c is 8, condition false, jumping to

|  |  |
| --- | --- |
| Line 14. | }else{ |

it is an else block, going in…

|  |  |
| --- | --- |
| Line 15. | c-=3; |

c=c-3

c=8-3

c=5

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

8 > 0, condition true, going in..

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j=j\*2

j(byte) = 64(byte) \* 2(int)

j(byte) = 64(int) \* 2(int)

j(byte) = 128(int)

wait, byte can’t hold 128, lets find out what is going to be stored.

128%256

128

128 is greater than 127

128-256

-128

j(byte) = -128(int)

j(byte) = -128(byte)

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=5-(-437)/2\*3-(-128)

d=5-(-218)\*3-(-128)

d=5-(-654)-(-128)

d=659-(-128)

d=787

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+”3”+j

787%2+”3”+(-128)

1+”3”+(-128)

“13”+(-128)

“13-128”

Great, we got our 7th output!

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

Since when 5 is less than 3? Condition is false.

|  |  |
| --- | --- |
| Line 15. | c-=3; |

c=c-3

c=5-3

c=2

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

2 is greater than 0, right?

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j=j\*2

j=-128(byte) \* 2(int)

j=-128(int) \* 2(int)

j=-256(int)

Yes, you are right, a byte cannot hold -256.

-256 % 256

0 (that was easy!)

j=0(int)

j=0(byte)

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=2-787/2\*3-0

d=2-393\*3-0

d=2-1179-0

d=-1177-0

d=-1177

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+”3”+j

-1177%2+”3”+0

-1+”3”+0

”-13”+0

”-130”

See! Now that you understand it, calculations are getting simpler!

Got 8th output.

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

c is 2. You do know that 2 is less than 3, correct?

|  |  |
| --- | --- |
| Line 13. | c--; |

c-- is a short hand for c=c-1;

c=c-1

c=2-1

c=1

We are done with if. So, we won’t enter the else block this time.

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

c is 1. What? We are still stuck in this loop just for 1? Going in for the last time.. It does not matter whether you understand it or not, I am not going to enter this loop again.

Whats next?

|  |  |
| --- | --- |
| Line 9. | j\*=2; |

j=j\*2

j=0\*2

j=0 (that’s why I love tracing!)

|  |  |
| --- | --- |
| Line 10. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=1-(-1177)/2\*3-0

d=1-(-588)\*3-0

d=1-(-1764)-0

d=1765-0

d=1765

|  |  |
| --- | --- |
| Line 11. | System.out.println(d%2+"3"+j); |

d%2+”3”+j

1765%2+”3” +0

1+”3” +0

“13” +0

“130”

Okay, 9th output.

|  |  |
| --- | --- |
| Line 12. | if (c<3){ |

1 is less than 3, going in..

|  |  |
| --- | --- |
| Line 13. | c--; |

c=c-1

c=1-1

c=0

|  |  |
| --- | --- |
| Line 8. | while (c>0){ |

Is zero greater than zero? NO. No number is greater than itself. It is equal to itself. Condition false. We are going in. We will jump out of the loop, to after, line 17.

|  |  |
| --- | --- |
| Line 18. | d=c-d/2\*3-j; |

d=c-d/2\*3-j

d=0-1765/2\*3-0

d=0-882\*3-0

d=0-2646-0

d=-2646-0

d=-2646

|  |  |
| --- | --- |
| Line 19. | System.out.println(d%2+"3"+j); |

d%2+”3”+j

-2646%2+”3”+0

0+”3”+0

”03”+0

”030”

You are a genius! You found the 10th output!

Writing this document was completely unnecessary because this quiz was based on past lab(s) and theory class. Using a bit of experiment with byte, everyone should be able to do it!