# CMSC 150
# INTRODUCTION TO COMPUTING

LAB – WEEK 2

- BINARY
- PROGRAMMING WITH DATA

# DECIMAL NUMBER REPRESENTATION

- Lets look at a number we are familiar with 9801. The most common number in the universe.

| 9 | 8 | 0 | 1 |
|---|---|---|---|
| $10^3 = 1000$ | $10^2 = 100$ | $10^1 = 10$ | $10^0 = 1$ |
| $9 * 10^3 +$ | $8 * 10^2 +$ | $0 * 10^1 +$ | $1 * 10^0 = 9801$ |

- When we add, we never let a digit be more than 9. Example: 9801 + 9

| 9 | 8 | 0 | 1 |
|---|---|---|---|
| + | | | 9 |
| 9 | 8 | 1 | 0 |

# BINARY NUMBER REPRESENTATION

- Synonymously, binary numbers work the same way. Except instead of base 10, it is base 2. A digit can only be 0 or 1. Example: 0010 0101

| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| $2^7 = 128$ | $2^6 = 64$ | $2^5 = 32$ | $2^4 = 16$ | $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^1 = 1$ |
| $0 * 2^7 +$ | $0 * 2^6 +$ | $1 * 2^5 +$ | $0 * 2^4 +$ | $0 * 2^3 +$ | $1 * 2^2 +$ | $0 * 2^1 +$ | $1 * 2^1 = 37$ |

- And adding 0010 1010 + 0000 0101

| | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|
| + | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

- Note there are other common number systems: Octal (base 8, digits 0-7) and Hexadecimal (base 16, digits 0-9 and A-F, used for memory addresses)

# ACTIVITY

- With a partner
- Convert the binary number 1001 1001 to decimal
- Add the binary number 0101 0101 to 1001 1001 (DO NOT DO THIS IN DECIMAL) and then convert to a decimal number
- Bonus: 0xA1 to decimal, add 0x0E to it and convert to decimal. Hint: 0x means that the number is a hexadecimal (base 16)

# SO WHAT DOES THIS MEAN?

- The amount of bits (like digits but only 2 options) defines the range of data
  - With 8 bits, like the previous slide, the range of an integer would be 0 – 255 or 0000 0000 – 1111 1111
- Also how in the world do we do negative numbers?
  - The "most significant bit" (one furthest left, or the one with the largest value, e.g., $2^7$ in our example) will define positive (0) or negative (1).
  - This allows the range to be -128 – 127
- So hopefully you see if we want to represent a number in the thousands we cannot use 8 bits….
- Note for an $n$ bit integer (signed) will have the range $[-2^n, 2^n - 1]$ and an $n$ bit integer (unsigned) will have the range $[0, 2^n - 1]$

# IN JAVA

- Java allows you to decide the size of your integers
  - **byte** – 8-bit signed, range $[-2^7, 2^7 - 1] = [-128, 127]$
  - **short** – 16-bit signed, range $[-2^{15}, 2^{15} - 1] = [-32768, 32767]$
  - **int** – 32-bit signed, range $[-2^{31}, 2^{31} - 1]$
  - **long** – 64-bit signed, range $[-2^{63}, 2^{63} - 1]$
- Some other types
  - **char** – 16-bit unsigned, range $[0, 2^{16} - 1]$, representing a single Unicode character
  - **float** – 32-bit floating point, accurate to about 7 significant digits
  - **double** – 64-bit floating point, accurate to about 15 significant digits

# PROGRAMMING WITH DATA

GET YOUR COMPUTER READY FOR PROGRAMMING

MAKE A FILE Numbers.java

# PROGRAMMING WITH DATA

- Make a program which declares and defines two integers and outputs their summation

```
1.  /* Declare 2 integers and output their sum */
2.  public class Numbers {
3.      public static void main(String[] args) {
4.          int a = 5, b = 7;
5.          System.out.println("Sum of " + a + " and " + b + " is " + (a + b));
6.      }
7.  }
```

Compile: javac Numbers.java

Run: java Numbers

Note the automatic type conversion of a, b, and (a+b) to Strings…

Try a+b without the ()…what happens?

# PROGRAMMING WITH DATA

- Make the program more interesting by allowing users to define their own numbers. Use command line arguments

```
1.  /* Input 2 integers and output their sum */
2.  public class Numbers {
3.      public static void main(String[] args) {
4.          int a = Integer.parseInt(args[0]), b = Integer.parseInt(args[1]);
5.          System.out.println("Sum of " + a + " and " + b + " is " + (a + b));
6.      }
7.  }
```

Integer.parseInt() converts a string to an integer

Compile: javac Numbers.java

Run: java Numbers 5 7

Try: java Numbers 1.5 7 and java Numbers 2. What happened?

# PROGRAMMING WITH DATA

- Allow data with decimals

```
1. /* Input 2 doubles and output their sum */
2. public class Numbers {
3.     public static void main(String[] args) {
4.         double a = Double.parseDouble(args[0]), b = Double.parseDouble(args[1]);
5.         System.out.println("Sum of " + a + " and " + b + " is " + (a + b));
6.     }
7. }
```

Double.parseDouble() converts a string to a double

Compile: javac Numbers.java

Run: java Numbers 1.5 7.8. This fixes one of our prior problems!

# PROGRAMMING WITH DATA

- Lets add a third parameter like this:

```
1. /* Input 2 doubles and output their sum */
2. public class Numbers {
3.     public static void main(String[] args) {
4.         double a = Double.parseDouble(args[0]), b = Double.parseDouble(args[1]);
5.         System.out.println("Sum of " + a + ", " + b + ", and " + c + " is " + (a + b + c));
6.     }
7. }
```

Compile: javac Numbers.java. Note the error!

# PROGRAMMING WITH DATA

- Lets mistakenly make c a boolean:

```
1. /* Input 2 doubles and output their sum */
2. public class Numbers {
3.     public static void main(String[] args) {
4.         double a = Double.parseDouble(args[0]), b = Double.parseDouble(args[1]);
5.         boolean c;
6.         System.out.println("Sum of " + a + ", " + b + ", and " + c + " is " + (a + b + c));
7.     }
8. }
```

Compile: javac Numbers.java. Note the error!

# PROGRAMMING WITH DATA

- Lets fix c to be a double:

```
1. /* Input 2 doubles and output their sum */
2. public class Numbers {
3.     public static void main(String[] args) {
4.         double a = Double.parseDouble(args[0]), b = Double.parseDouble(args[1]);
5.         double c;
6.         System.out.println("Sum of " + a + ", " + b + ", and " + c + " is " + (a + b + c));
7.     }
8. }
```

Compile: javac Numbers.java. Note the error!

# PROGRAMMING WITH DATA

- Lets now make a correct initialization of c to fix the program:

```
1. /* Input 3 doubles and output their sum */
2. public class Numbers {
3.     public static void main(String[] args) {
4.         double a = Double.parseDouble(args[0]), b = Double.parseDouble(args[1]);
5.         double c = Double.parseDouble(args[2]);
6.         System.out.println("Sum of " + a + ", " + b + ", and " + c + " is " + (a + b + c));
7.     }
8. }
```

Compile: javac Numbers.java

Run: java Numbers 1.5 4.3 2.3

# PROGRAMMING WITH DATA

- Lets explore precedence:

```
1.  /* Input 3 doubles and output their sum */
2.  public class Numbers {
3.      public static void main(String[] args) {
4.          double a = Double.parseDouble(args[0]), b = Double.parseDouble(args[1]);
5.          double c = Double.parseDouble(args[2]);
6.          System.out.println("Result of " + a + " + " + b + " * " + c + " is " + (a + b * c));
7.          System.out.println("Result of (" + a + " + " + b + ") * " + c + " is " + ((a + b) * c));
8.      }
9.  }
```

Compile: javac Numbers.java

Run: java Numbers 1.5 4.3 2.3

# PROGRAMMING WITH DATA

- Lets explore integer division and modulus (remainder). Change to 2 integer parameters and compute different equations.

```
1.  /* Input 2 integers to explore / and % */
2.  public class Numbers {
3.      public static void main(String[] args) {
4.          int a = Integer.parseInt(args[0]), b = Integer.parseInt(args[1]);
5.          System.out.println("Result of " + a + " / " + b + " is " + (a / b));
6.          System.out.println("Result of " + a + " % " + b + " is " + (a % b));
7.      }
8.  }
```

Compile: javac Numbers.java

Run: java Numbers 15 3 and java Numbers 17 3

# PROGRAMMING WITH DATA

- Lets explore equality and Booleans

```
1.  /* Input 2 integers and determine if they are equal to each other */
2.  public class Numbers {
3.      public static void main(String[] args) {
4.          int a = Integer.parseInt(args[0]), b = Integer.parseInt(args[1]);
5.          boolean c = a == b;
6.          System.out.println("Result of " + a + " == " + b + " is " + c);
7.          System.out.println("Result of a = b is " + (a = b));
8.      }
9.  }
```

Compile: javac Numbers.java

Run: java Numbers 5 3 and java Numbers 3 3

Note that there is a big difference between assignment (=) and equality checking (==)

# EXERCISES – WITH A PARTNER

1. Try and find a weird quirk of java types, either in compiler errors, precedence, or strange operators
2. Make a program to compute the distance between two points. Each point has an x and y component. Recall the formula $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Program first with int and then with double. What differences did you notice?
3. Program a guessing game – Takes as input N (the high number limit) and the guess. So generate a random number between 1 and N and compare to the guess.
4. Work on programming assignment 2. This is solitary, not with a partner.