CS251L

REVIEW

2010.8.25 | Derek Trumbo | UNM

Java Applications

- Java application defined by a Java class with a main method
  - `public static void main(String[] args)`
    - `args` is an array of strings represented the command line parameters passed to the application
  - The public class must match the name of the file

Java Applications

- Though usually hidden when using an IDE, know that the “javac” command compiles .java files, and the “java” command executes the resulting Java applications

  ```
  cd MyCode/
  javac MyApp.java
  java MyApp
  ```

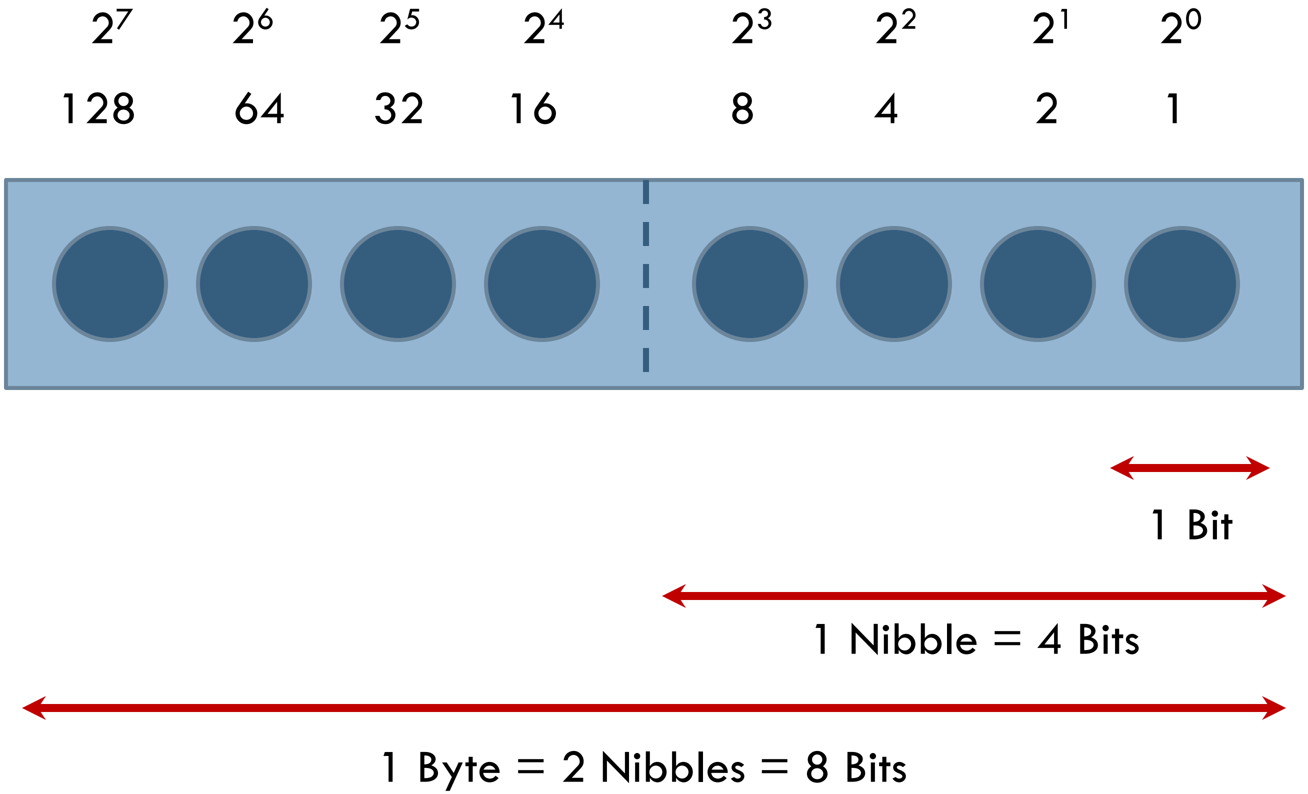
Java Applications

- Whereas historically most programming languages have been designed to be completely compiled (C, C++) or completely interpreted (Perl, Python, JavaScript), Java is both compiled, and then interpreted
- The "javac" command compiles Java code into "bytecode" and then the "java" command interprets this bytecode
  - Eclipse executes both of these for you "under the hood"
- One seminal goal of Java was platform independence

Data Types

- Understanding data types foundation of all programming
- Two general categories in any language:
  - Primitive data types
  - Abstract data types (classes)
- Not all programming languages have the exact same primitive data types, but the overlap is large among compiled languages

The Byte



2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 64 32 16 8 4 2 1

1 Bit

1 Nibble = 4 Bits

1 Byte = 2 Nibbles = 8 Bits

Unsigned Byte

Min Value:

0 (all off)

Max Value:

255 (all on)

Total # Possible Values:

256 = 2^8

Signed Byte

Min Value:

-128

Max Value:

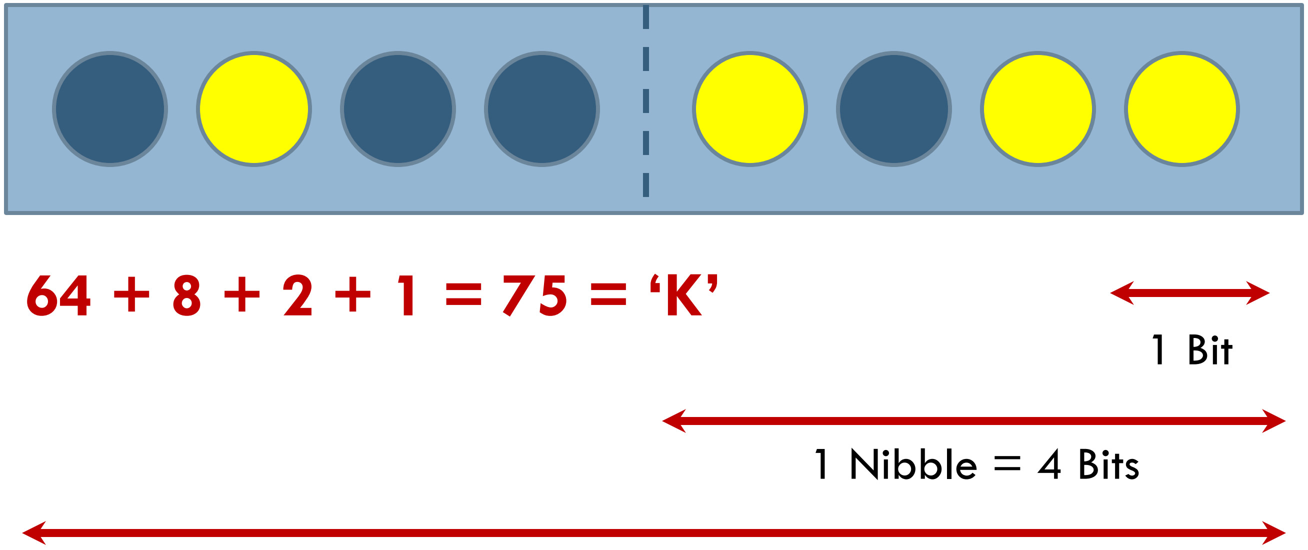
127

Total # Possible Values:

256 still

The Byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



64 + 8 + 2 + 1 = 75 = ‘K’

1 Bit

1 Nibble = 4 Bits

1 Byte = 2 Nibbles = 8 Bits

Unsigned Byte

Min Value:

0 (all off)

Max Value:

255 (all on)

Total # Possible Values:

256 = 2^8

Signed Byte

Min Value:

-128

Max Value:

127

Total # Possible Values:

256 still

Primitive Data Types

- boolean [1 bit]: true, false
- byte [8 bits]: -128 to 127 (rarely used)
- char [16 bits]: 0 to 65,535 (e.g. ‘a’, ‘B’, ‘$’, ‘7’)
- short [16 bits]: -32,768 to 32,767 (rarely used)
- int [32 bits]: -2,147,483,648 to 2,147,483,647
- long [64 bits]: -9.2 x 10^{18} to 9.2 x 10^{18} (approx)
- float [32 bits]: -1.4 x 10^{-45} to 3.4 x 10^{38} (approx)
- double [64 bits]: -4.9 x 10^{-324} to 1.8 x 10^{308} (apx)

Operators

- Arithmetic (+ - * / %)
- Relational (< <= > >=)
- Equality (== !=)
- Logical (&& ||)
- Bitwise (<< >> >>> & ^ |)
- Assignment (= += -= *= /= etc.)
- Others (?: ++ -- etc.)

Operator Precedence

- Just like in math, certain operators execute before others (A + B * C)
- Refer to this table for precedence:
  - http://download.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Operator Associativity

- What happens when multiple operators at the same level of precedence exist in sequence in an expression is defined by associativity
- Operators either “associate” left-to-right or right-to-left; most associate left-to-right

Operator Associativity

- 10 + 20 + 3 * 4 * 5 - 30
- 10 + 20 + 12 * 5 - 30
- 10 + 20 + 60 - 30
- 30 + 60 - 30
- 90 - 30
- 60

Operator Associativity

□ var1 = var2 = var3 = 0;

□ var1 = var2 = 0;

□ var1 = 0;

□ 0; !!theSame

□ The equality operators associate right-to-left

□ Not only does the equality operator assign a value to the variable, it returns the value for subsequent expressions

Flow Control Statements

- Decision
  - if-else
  - switch (implemented in PL’s as a convenience)
- Iteration
  - for (counted loop)
  - while (top-tested loop)
  - do-while (bottom-tested loop)

Decision Statements

- if-else statements

```
if(x > 10)
    doSomething();

if(x > 10) {
    doSomething();
}

if(x > 10) {
    doSomething();
    doSomethingElse();
}
```

Decision Statements

- if-else statements

```
if(x > 10)
   doSomething();
   doSomethingElse();
```

- No-no! Don’t confuse yourself – if you leave off the braces only the first statement will be in the if and the second statement will always be executed no matter what

Decision Statements

- if-else statements

```
if(x > 10) {
   doSomething();
} else {
   doSomethingElse();
}

if(x > 10) {
   doSomething10();
} else if(x > 5) {
   doSomething5();
} else {
   doSomethingElse();   // Executed when x <= 5
}
```

Decision Statements

- switch statements (used only with primitives)

```
switch(myInt) {
   case 1: doOne(); break;
   case 2: doTwo(); break;
   default: doOtherwise(); break;
}
// Used in place of this:
if(myInt == 1) {
   doOne();
} else if(myInt == 2) {
   doTwo();
} else {
   doOtherwise();
}
```

Iteration

- for loops
  - When you know exactly how many times you want some piece of code to execute

```
for(initialization; condition; inc/dec) {
   // loop body
}
```

Iteration

```
for(int e = 1; e <= 10; e++)
   sum += e;

for(int h = 100; h >= 0; h--) {
   System.out.println("height = " + h);
}

int count = 0;
for(double d = -2.3; d <= 293.48; d += 5.66) {
   count++;
   System.out.println(d + " " + count);
}
```