CMSC 131

Object-Oriented Programming I

Bitwise Operations

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Overview

- Bitwise Operators
- BitSet class

Operators Revisited

- **Operators**: We discussed various operators (+, -, *, <, ==, &&, ||) earlier this semester, but omitted some:

  Bitwise operators: Operate on values as **binary numbers**

- **Bitwise Operators**: Recall that all quantities are stored as binary numbers in memory. For example:

```
int x = 1037;       // binary: ...0010000001101 filled out to 32 bits
char c = 'y';       // binary: ...0000001111001 filled out to 16 bits
boolean b = true;   // binary: 1
```

- You are **not** required to know how these conversions are performed. (It is covered in later courses.)
- Java's bitwise operators act on these binary representations

Bitwise Operators

- Java supports the standard bit operators:

  ~a: **complement** of a

  a & b: **and** (1 if **both** a and b are 1)

  a | b: **or** (1 if **either** a and b are 1)

  a ^ b: **exclusive or** (1 if **either** a or b is 1, but **not both**)

a	b	~a	a & b	a | b	a ^ b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Bitwise Operators

- Java's bitwise operators can be applied
  - to any **integral type**: **char**, **byte**, **short**, **int**, **long**
  - to **boolean**
- When applied to integral types, the operations are applied **bitwise**:

```
int a = 45;      // a = ...00101101
int b = 14;      // b = ...00001110
int c = a & b;   // c = (00101101 & 00001110) = 00001100 (= 12)
```

00101101	00101101	00101101
& 00001110	| 00001110	^ 00001110
00001100	00101111	00100011

- **Who uses these**: They are used in often hardware-related tasks (device management) and have other surprising uses. (E.g.: Using exclusive-or you can swap to integers without a temporary.

Bitwise Operators

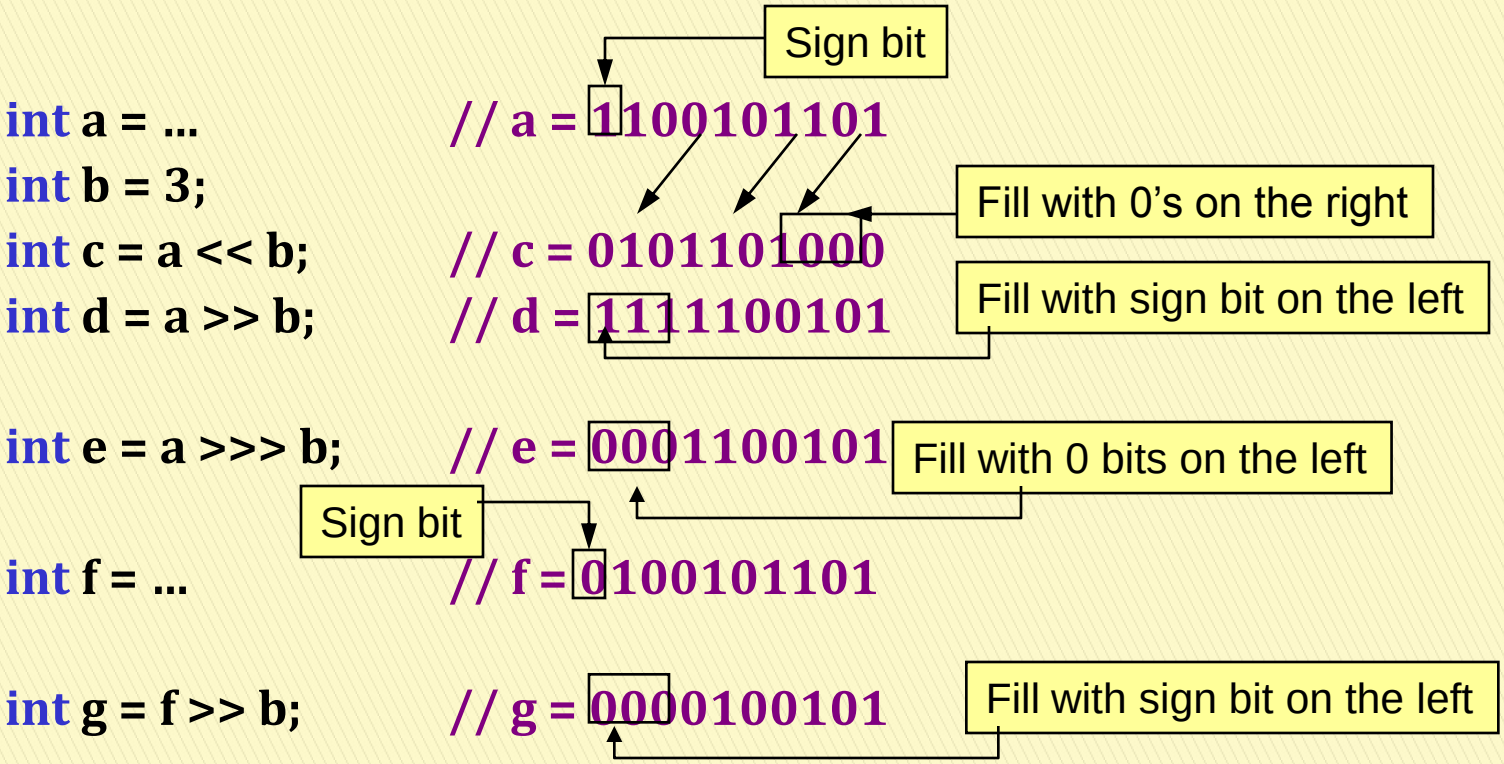
- **Who uses these**:
  - They are used in often hardware-related tasks (device management) and have
  - To multiply shift left by powers of two
  - To divide shift right by powers of two
  - Other surprising uses. (E.g.: Using exclusive-or you can swap to integers without a temporary)
- **Using XOR to swap without temporary**
  - A = A **XOR** B
  - B = A **XOR** B
  - A = A **XOR** B
- **Example:**
  http://en.wikipedia.org/wiki/XOR_swap_algorithm

Shift Operators

- Another common operation involves **shifting** bits left or right
  - **a << b**: Shift **a** **left** by **b** positions (filling with 0's)
  - **a >> b**: Shift **a** **right** by **b** positions (filling with the **sign bit**)
  - **a >>> b**: Shift **a** **right** by **b** positions (filling with **0**'s)
- **Notes**:
  - **a** must be **integral** type (byte, short, …, long).
  - **b** should be a **nonnegative integral** type
- **Sign bit**: Because there is no "-" sign in binary, Java encodes negative numbers using a method called **2's-complement representation**. We will not discuss this, but a key element is that the leftmost bit, called the **sign bit**, is:
  - 0 for **positive** numbers
  - 1 for **negative** numbers
- We often want to keep the sign bit **unchanged** when shifting

Shift Operators

- **Example**: Rather than use 32-bit int's, we use a 10-bit example



BitSet Class

- The BitSet class implements a vector of bits

http://download.oracle.com/javase/6/docs/api/java/util/BitSet.html

- Let's take a look at the class methods
- How can we use this class in order to implement a Sudoku validator?