Using the Unary Operators

- An operator that requires one operand is called a unary operator.
- The types of unary operators:

Unary Plus operator (+)	• Indicates a positive value.
Unary Minus operator (−)	• Indicates a negative value.
Increment operator (++)	• Increments the value by 1.
Decrement operator (−−)	• Decrements the value by 1.
Logical Complement operator (!)	• Inverts the value.

Using the Unary Operators (Contd.)

- Play the animation, which explains some of the concept of unary operators.

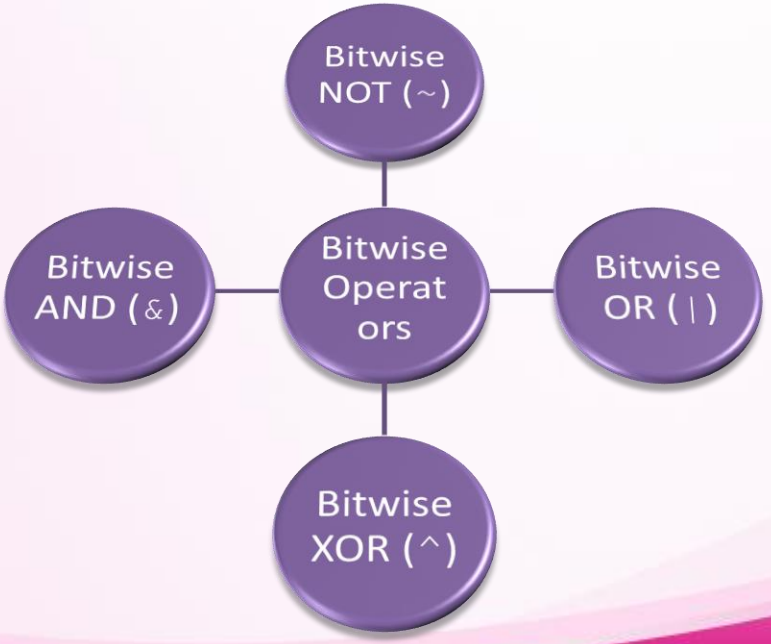
Use the UnaryOperators animation

Using the Unary Operators (Contd.)

- The increment and decrement operators can be applied in the prefix and postfix forms.
- In the prefix form:
  - The operator precedes the operand.
  - The value is incremented or decremented before it is assigned to the operand.
- In the postfix form:
  - The operator follows the operand.
  - The value is incremented or decremented after it has been assigned to the operand.

Using the Bitwise Operators

- Bitwise operators are used:
  - For the manipulation of data at the bit level.
  - To operate on the individual bits of their operands.
- The operands can be of various data types, such as `int`, `short`, `long`, `char`, and `byte`.
- Bitwise operators can be categorized into:



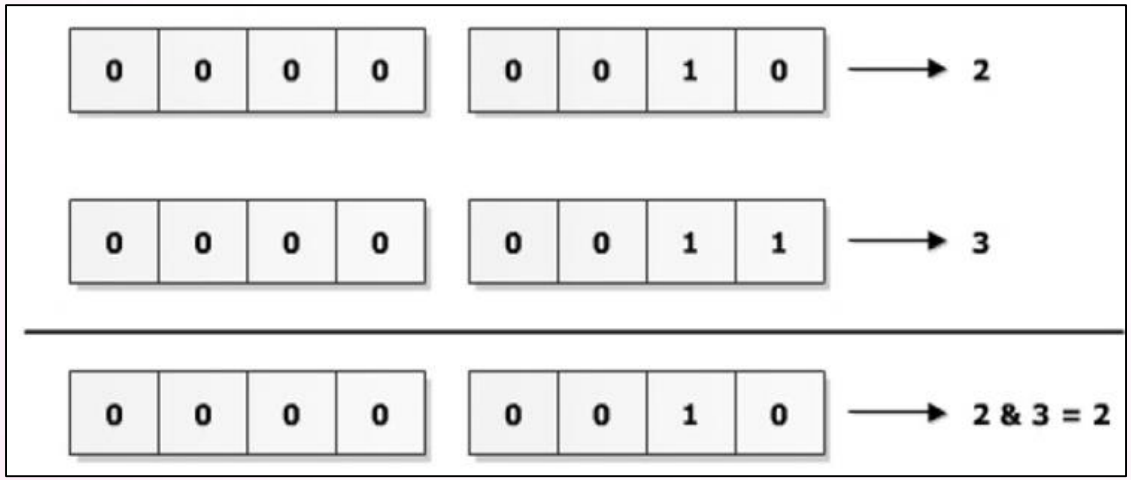
Using the Bitwise Operators (Contd.)

- Bitwise AND operator:
  - Performs an AND operation on two operands.
  - Results `1` if both bits are `1` else `0` in all other cases.
- The following table lists the bitwise AND operation.

Operand 1	Operand 2	Operand 1 & Operand 2
0	0	0
0	1	0
1	0	0
1	1	1

Using the Bitwise Operators (Contd.)

- The following figure displays the bitwise AND operation for `2` and `3`.



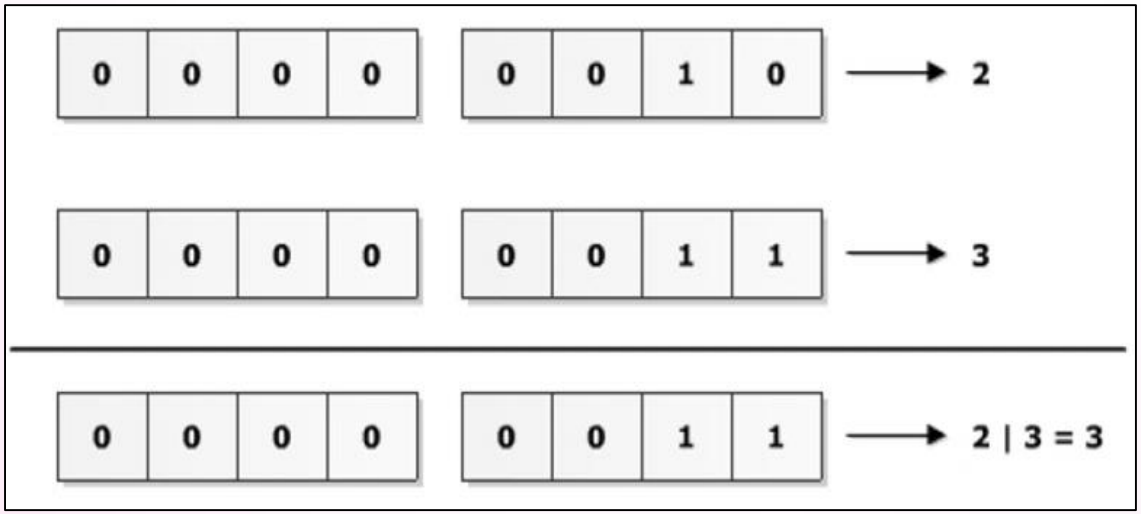
Using the Bitwise Operators (Contd.)

- Bitwise OR operator:
  - Performs the OR operation on two operands.
  - Results, `0`, if both bits are `0`, else `1` in all other cases.
- The following table lists the bitwise OR operation.

Operand 1	Operand 2	Operand 1 | Operand 2
0	0	0
0	1	1
1	0	1
1	1	1

Using the Bitwise Operators (Contd.)

- The following figure displays the bitwise OR operation for `2` and `3`.



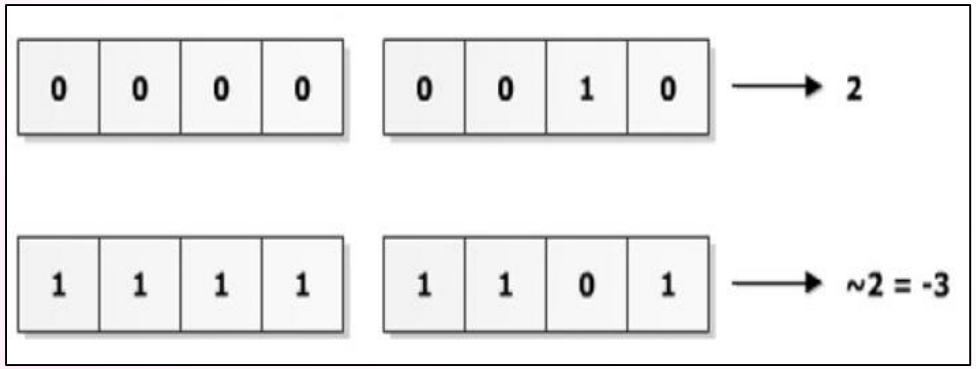
Using the Bitwise Operators (Contd.)

- Bitwise NOT operator:
  - Is a unary operator.
  - Is used to performs the NOT operation on each bit of a binary number.
  - Is also called Bitwise complement.
  - Is used to invert or complement each of the bits of a binary number.
- The following table lists the bitwise NOT operation.

Operand 1	~Operand 2
0	1
1	0

Using the Bitwise Operators (Contd.)

- The following figure displays the bitwise NOT operation for `2`.



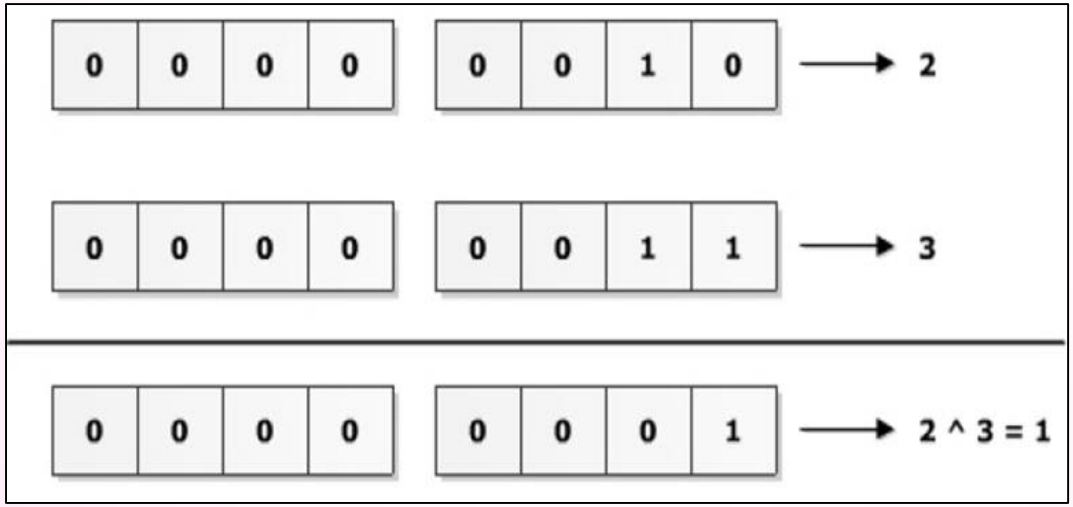
Using the Bitwise Operators (Contd.)

- Bitwise XOR operator:
  - Performs the XOR operation on two operands.
  - Results in `1`, if exactly one bit is `1`, else `0` in all other cases.
- The following table lists the bitwise XOR operation.

Operand 1	Operand 2	Operand 1 ^ Operand 2
0	0	0
0	1	1
1	0	1
1	1	0

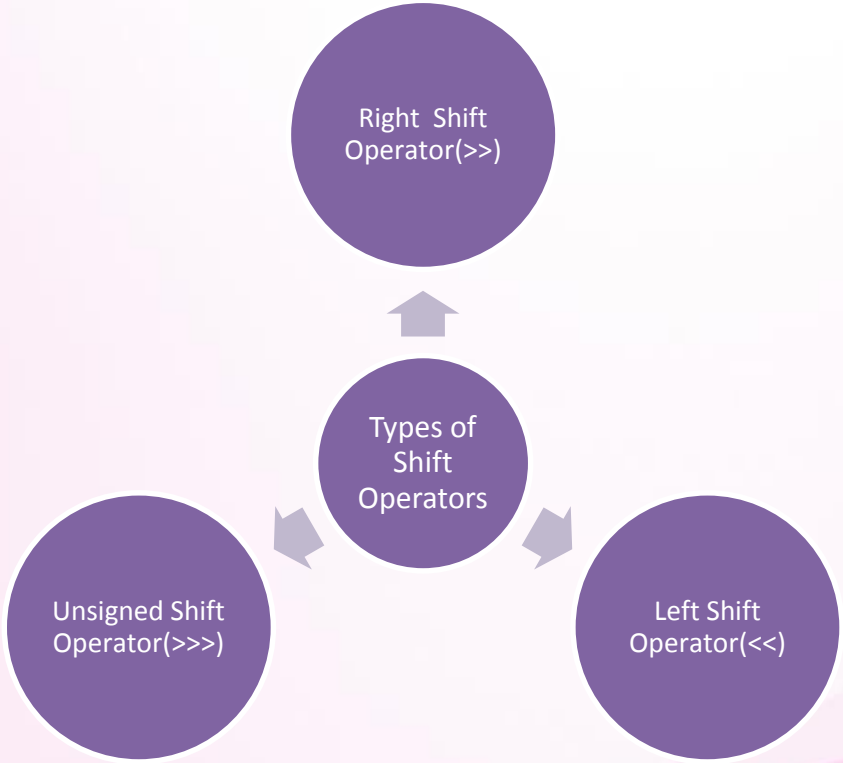
Using the Bitwise Operators (Contd.)

- The following figure displays the bitwise XOR operation for 2 and 3.



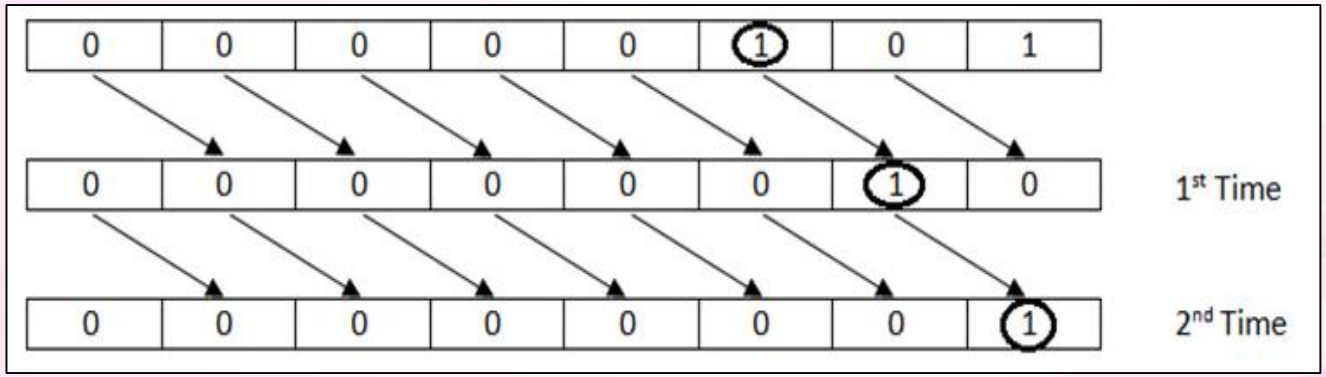
Using the Shift Operators

- A shift operator is used to shift the bits of its operand either to the left or to the right.
- The various types of shift operators:



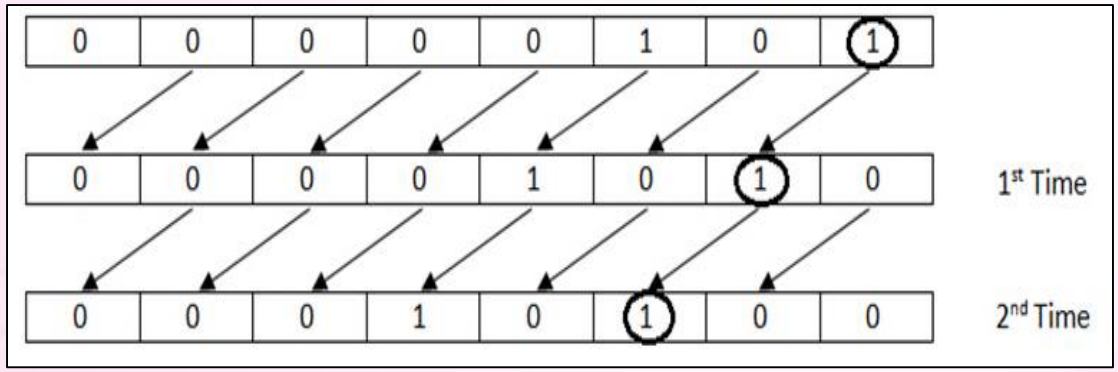
Using the Shift Operators (Contd.)

- Right shift operator:
  - Shifts all the bits of a binary number in the right direction.
  - The syntax for the right shift operator is:
    `operand >> num`
- The following figure displays the right shift operation of the `5>>2` expression.



Using the Shift Operators (Contd.)

- Left shift operator:
  - Shifts all the bits of a binary number in the left direction.
  - The syntax for the left shift operator is:
    `operand << num`
- The following figure displays the left shift operation of the `5<<2` expression.



Using the Shift Operators (Contd.)

- Unsigned right shift operator:
  - Shift the bits of a binary number to the right.
  - Fills the leftmost bits of a binary value with `0`, irrespective of whether the number has `0` or `1` at the leftmost bit.
  - Is generally used with the `32` and `64` bit binary numbers.

Using the Ternary Operator

- Ternary operator is used to:
  - Evaluate an expression.
  - Work on a logical expression and two operands.
- The syntax of ternary operator is:

```
boolean_expression ? expression 1 : expression 2
```

- It returns one of the two operands depending on the result of the expression.

Just a minute

- Which one of the following operators is used to find the remainder and assign the result to a variable?
  - %
  - %=
  - /
  - /=

Just a minute (Contd.)

- Solution:
  - %=