B.Sc (Prog)
Programming in JAVA
SEM V

19-08-2020

- Java programs are a collection of whitespace, identifiers, literals, comments, operators, separators, and keywords
- Java is a free-form language. This means that you do not need to follow any special indentation rules.

Identifiers

- Identifiers are used to name things, such as classes, variables, and methods

<u>Rules for identifier</u>

- An identifier may be any descriptive sequence of uppercase and lowercase letters, numbers, or the underscore and dollar-sign characters.
- must not begin with a number, lest they be confused with a numeric literal

Identify valid and invalid identifier

- AvgTemp
- count
- A4
- $test
- this_is_ok
- 2count
- high-temp
- Not/ok

Literals

- A constant value in Java is created by using a *literal representation of it.*
- *int a="hello"*

hello is literal here

a is identifier

Comments

- Single line comment

Starts with //

- Multiline comment

start with /* and ends with */

- Documentation comment

Used to used to produce an HTML file that documents your program. Documentation comment begins with a /** and ends with a */

Operators in JAVA

divided into the following four groups:

1. Arithmetic
2. Bitwise
3. Relational
4. logical

Arithmetic operators

- used in mathematical expressions
- The operands of the arithmetic operators must be of a numeric type.
- cannot use them on **boolean types, but arithmetic operator can be performed on char types**

```
class BasicMath {
public static void main(String args[]) {
System.out.println("Integer Arithmetic");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c - a;
int e = -d;
System.out.println("a = " + a);     System.out.println("b = " + b);
System.out.println("c = " + c);      System.out.println("d = " + d);
System.out.println("e = " + e);
System.out.println("\nFloating Point Arithmetic");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc - a;
double de = -dd;
System.out.println("da = " + da);
System.out.println("db = " + db);
System.out.println("dc = " + dc);
System.out.println("dd = " + dd);
System.out.println("de = " + de);
} }
```

Arithmetic Compound Assignment Operators

- a = a + 4; can be written this statement as

a += 4;

+= *compound assignment operator*

var = var op expression;

can be rewritten as   *var op= expression;*

Increment and Decrement operator

- In the prefix form, the operand is incremented or decremented before the value is obtained for use in the expression.
- In postfix form, the previous value is obtained for use in the expression, and then the operand is modified.

```
int a = 1;
int b = 2;
int c;
int d;
c = ++b;
d = a++;
c++;
```

OUTPUT

```
a = 2
b = 3
c = 4
d = 1
```

Bitwise Logical Operators

The bitwise logical operators are **&, |, ^, and ~**

bitwise complement, the unary NOT operator, ~, inverts all of the bits of its operand.

For example, the number 42, which has the following bit pattern:

00101010   becomes 11010101

The Bitwise AND

The AND operator, **&, produces a 1 bit if both operands are also 1. A zero is produced in** all other cases.

Here is an example:

00101010 42

&00001111 15

____

00001010 10

The Bitwise OR

The OR operator, **|, combines bits such that if either of the bits in the operands is a 1, then** the resultant bit is a 1, as shown here:

00101010 42

| 00001111 15

____

00101111 47

The Bitwise XOR

The XOR operator, **^, combines bits such that if exactly one operand is 1, then the result** is 1. Oth erwise, the result is zero.

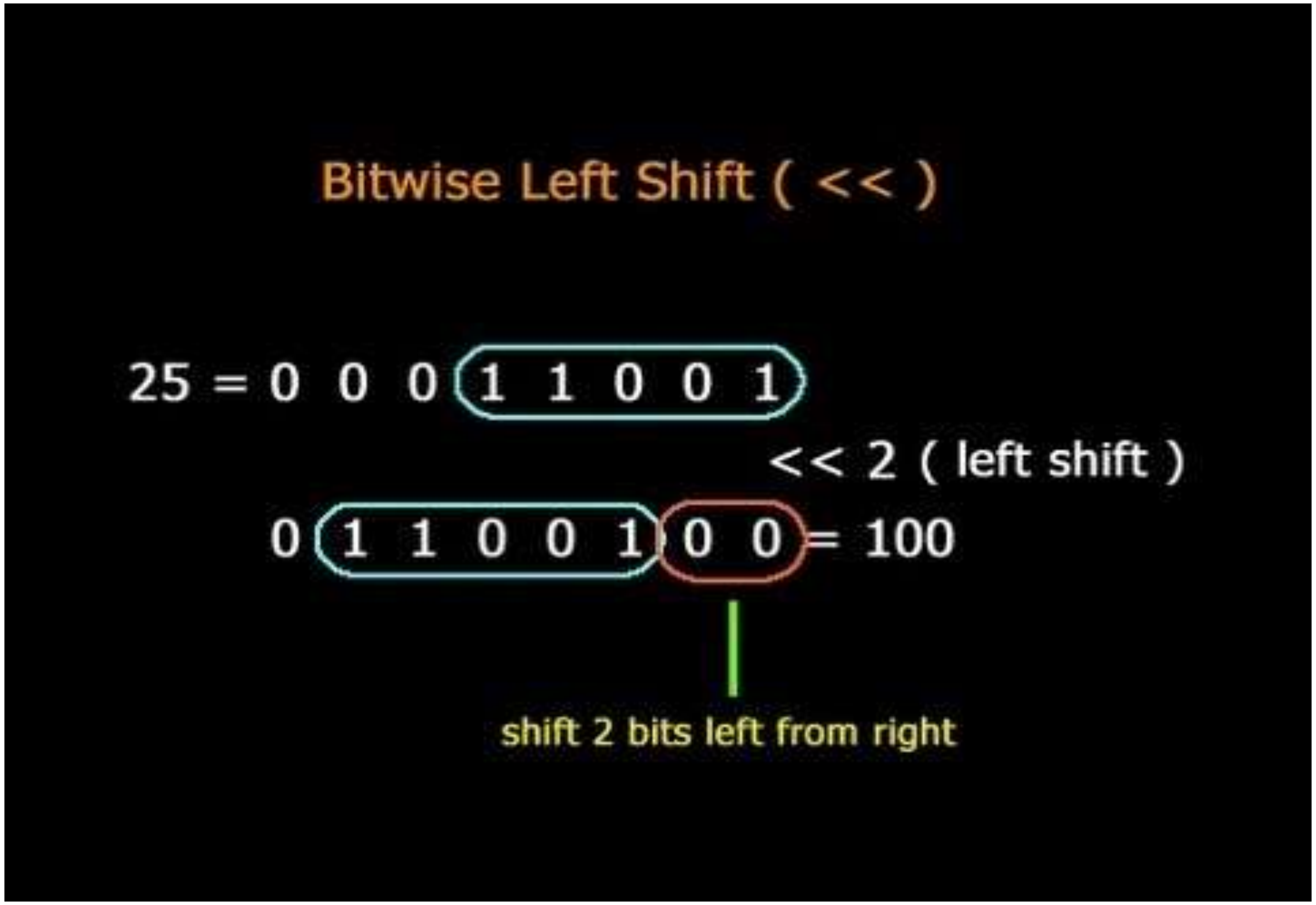
Example

```
  00101010      42
^ 00001111      15
__________
  00100101      37
```

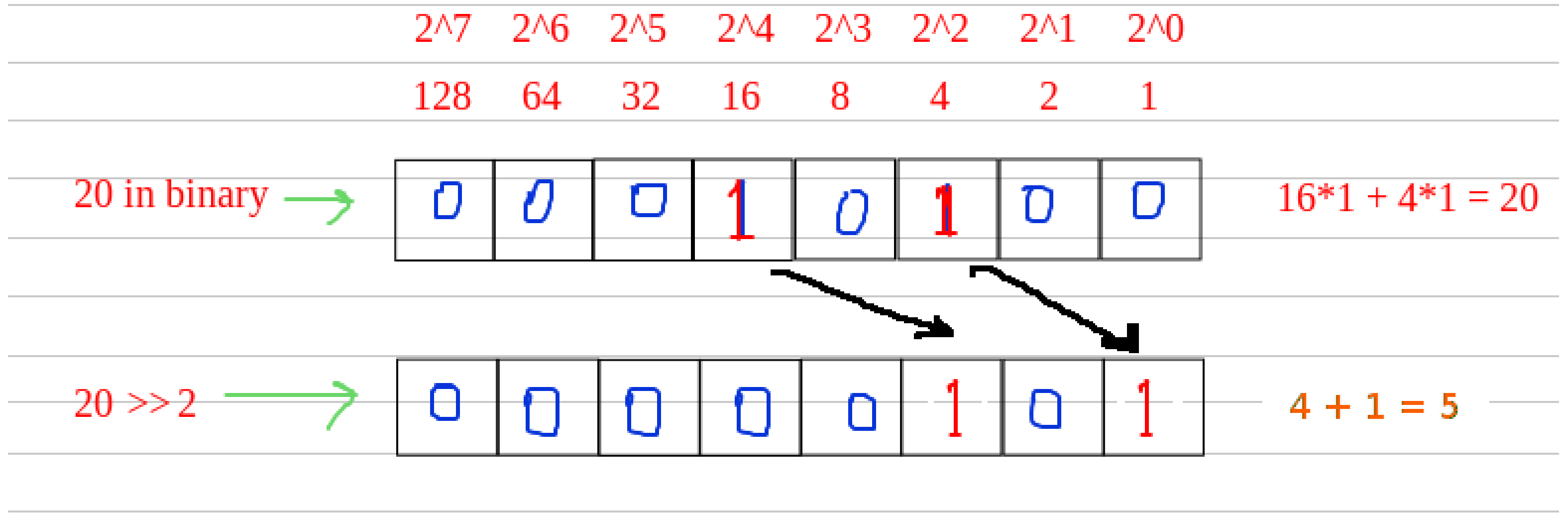
Calculate value of c,d,e,f

```
int a = 3;
int b = 6;
int c = a | b;
int d = a & b;
int e = a ^ b;
int f = (~a & b)|(a & ~b);
```

Left shift


Bitwise Left Shift (<<)
25 = 0 0 0 1 1 0 0 1
<< 2 (left shift)
0 1 1 0 0 1 0 0 = 100
shift 2 bits left from right

Right Shift (>>)



	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	128	64	32	16	8	4	2	1	
20 in binary →	0	0	0	1	0	1	0	0	16*1 + 4*1 = 20
20 >> 2 →	0	0	0	0	0	1	0	1	4 + 1 = 5

- left shift (<<)

X<<n gives x=x*2^n

- Right shift(>>)

X>>n gives x=x/2^n

Exercise

- What is the result after execution of following expressions in java ?

```
(i) int n=4, m=6, p=5;
n+=m%p+2
(ii) int p=2, n=4;
int k=n&lt;&lt;p;
```

- Give ouput of following code:

```
Class A
{
public static void main(String args[]){
int i1=5;
int i2=6;
String s1=”7”;
system.out.println(s1+i1+i2);
system.out.println(s1+i1+i2);
}}
```