

Chapter 2 Primitive Data Types and Operations

- Valid identifiers: applet, Applet, \$4, apps
Invalid identifiers: a++, --a, 4#R, #44
- Keywords:
class, public, int
- `int count = 100;`
`final int SIZE = 20;`
- There are three benefits of using constants: (1) you don't have to repeatedly type the same value; (2) the value can be changed in a single location, if necessary; (3) the program is easy to read.

Invalid identifiers: a++, --a, 4#R, #44

5.

`a = 46 / 9; => a = 5`

`a = 46 % 9 + 4 * 4 - 2; => a = 1 + 16 - 2 = 15`

`a = 45 + 43 % 5 * (23 * 3 % 2); => a = 45 + 3 * (1) = 48`

`a %= 3 / a + 3; => a %= 3 + 3; a % = 6 => a = a % 6 = 1;`

`d = 4 + d * d + 4; => 4 + 1.0 + 4 = 9.0`

`d += 1.5 * 3 + (++a); => d += 4.5 + 2; d += 6.5; => d = 7.5`

`d -= 1.5 * 3 + a++; => d -= 4.5 + 1; => d = 1 - 5.5 = -4.5`

- For byte, from -128 to 127, inclusive.
For short, from -32768 to 32767, inclusive.
For int, from -2147483648 to 2147483647, inclusive.
For long, from -9223372036854775808 to 9223372036854775807.
For float, the smallest positive float is 1.40129846432481707e-45 and the largest float is 3.40282346638528860e+38.
For double, the smallest positive double is 4.94065645841246544e-324 and the largest double is 1.79769313486231570e+308d.
- $25/4= 6$. If you want the quotient to be a floating-point number, rewrite it as $25.0/4.0$.
- Yes, the statements are correct. The printout is

the output for `25 / 4` is `6`;
the output for `25 / 4.0` is `6.25`;
- $4.0 / (3.0 * (r + 34)) - 9 * (a + b * c) + (3.0 + d * (2 + a)) / (a + b * d)$

10. b and c are true.
11. All.
12.
 - Line 2: Missing static for the main method.
 - Line 2: string should be String.
 - Line 3: i is defined but not initialized before it is used in Line 5.
 - Line 4: k is an int, cannot assign a double value to k.
 - Lines 7-8: The string cannot be broken into two lines.
13. Yes. Different types of numeric values can be used in the same computation through numeric conversions referred to as *casting*.
14. The fractional part is truncated. Casting does not change the variable being cast.
15.
 - f is 12.5
 - i is 12
16.

```
System.out.println( (int) '1' );
System.out.println( (int) 'A' );
System.out.println( (int) 'B' );
System.out.println( (int) 'a' );
System.out.println( (int) 'b' );

System.out.println( (char) 40 );
System.out.println( (char) 59 );
System.out.println( (char) 79 );
System.out.println( (char) 85 );
System.out.println( (char) 90 );

System.out.println( (char) 0X40 );
System.out.println( (char) 0X5A );
System.out.println( (char) 0X71 );
System.out.println( (char) 0X72 );
System.out.println( (char) 0X7A );
```
17. '\u345dE' is wrong. It must have exactly four hex numbers.
18. '\\' and '\"'
19.
 - i becomes 49, since the ASCII code of 'I' is 49;

j become 99 since (int)'1' is 49 and (int)'2' is 50;
k becomes 97 since the ASCII code of 'a' is 97;
c becomes character 'z' since (int) 'z' is 90;

20. <, <=, ==, !=, >, >=
21. (true) && (3 > 4)
false
- !(x > 0) && (x > 0)
false
- (x > 0) || (x < 0)
true
- (x != 0) || (x == 0)
true
- (x >= 0) || (x < 0)
true
- (x != 1) == !(x == 1)
true
22. (x > 1) && (x < 100)
23. ((x > 1) && (x < 100)) || (x < 0)
24. x > y > 0
incorrect
- x = y && y
incorrect
- x /= y
correct
- x or y
incorrect
- x and y
incorrect
25. The precedence order for boolean operators is &, ^, !, &&, and ||
true | true && false is false
- true || true && false is true

true | true & false is true.

26.

- a. The operands are evaluated first and from left to right. So $(--i + i + i++)$ is $-1 - 1 - 1$. $i++$ return the value of i , then i is incremented by 1. So, in the next `println` statement $i + ++i$ is $0 + 1$.
- b. The operands are evaluated first and from left to right. So, the left i before the $+$ operator is 0 and right i is assigned to 1 by $(i = 1)$. Therefore the `println` statement prints 1.
- c. The operands are evaluated first and from left to right. So, before i in the right of the $+$ operator is evaluated, i is assigned to 1. Therefore, $(i = 1) + i$ evaluates to 2.

27.

`a = (a = 3) + a; => a = 3 + 3 = 6`

`a = a + (a = 3); => a = 1 + 3 = 4`

`a += a + (a = 3); => the value of a in the left side of += is obtained (1), a + (a = 3) is 4, therefore, the final result a is 5.`

`a = 5 + 5 * 2 % a--; => a = 5 + 10 % a-- = 5 + 0 = 5`

`a = 4 + 1 + 4 * 5 % (++a + 1) => a = 5 + 4 * 5 % (++a + 1) = 5 + 20 % (++a + 1) = 5 + 20 % (2 + 1) = 5 + 20 % 3 = 5 + 2 = 7;`

`d += 1.5 * 3 + (++d); => the value of d in the left side of += is obtained (1.0), 1.5 * 3 + (++d) = 4.5 + (++d) = 4.5 + 2.0 = 6.5, therefore, the final result a is 7.5.`

`d -= 1.5 * 3 + d++; => the value of d in the left side of += is obtained (1.0), 1.5 * 3 + d++ = 4.5 + d++ = 4.5 + 1.0 = 5.5, therefore, the final result is 1.0 - 5.5 = -4.5.`

28.

`System.out.println("1" + 1); => 11`

`System.out.println('1' + 1); => 50` (since the Unicode for 1 is 49

`System.out.println("1" + 1 + 1); => 111`

`System.out.println("1" + (1 + 1)); => 12`

`System.out.println('1' + 1 + 1); => 51`

29.

Use `Double.parseDouble(string)` to convert a decimal string into a double value.

Use `Integer.parseInt(string)` to convert an integer string into an int value.

30.

`long totalMills = System.currentTimeMillis()` returns the milliseconds since Jan 1, 1970. `totalMills % (1000 * 60 * 60)` returns the current minute.

31.

The specifiers for outputting a boolean value, a character, a decimal integer, a floating-point number, and a string are `%b`, `%c`, `%d`, `%f`, and `%s`.

32.

- (a) the last item 3 does not have any specifier.

- (b) There is not enough items
- (c) The data for %f must a floating-point value

33.

- (a) amount is 32.320000 3.233000e+01
- (b) amount is 32.3200 3.2330e+01
- (c) *false // * denote a space
- (d) **Java // * denote a space

34.

Use // to denote a comment line, and use /* paragraph */ to denote a comment paragraph.

35.

Class names: Capitalize the first letter in each name.
Variables and method names: Lowercase the first word, capitalize the first letter in all subsequent words.
Constants: Capitalize all letters.

36.

```
public class Test {  
    /** Main method */  
    public static void main(String[] args) {  
        // Print a line  
        System.out.println("2 % 3 = " + 2 % 3);  
    }  
}
```

37.

Compilation errors are detected by compilers. Runtime errors occur during execution of the program. Logic errors results in incorrect results.

38.

```
1 + "Welcome " + 1 + 1 is 1Welcome 11.  
  
1 + "Welcome " + (1 + 1) is 1Welcome 2.  
  
1 + "Welcome " + ('\u0001' + 1) is 1Welcome 2  
  
1 + "Welcome " + 'a' + 1 is 1Welcome a1
```

39. The following conversions are not allowed:

```
char c = 'A';  
i = (int)c; // i becomes 65  
  
boolean b = true;  
i = (int)b; // Not allowed
```

```
float f = 1000.34f;
int i = (int)f; // i becomes 1000

double d = 1000.34;
int i = (int)d; // i becomes 1000

int i = 97;
char c = (char)i; // c becomes 'a'

int i = 1000;
boolean b = (boolean)i; // Not allowed
```

40.

x is 2.

41.

x is 1.

42.

d
d
false
-4