



Chapter 1

Data Types & Computation

TAO Yida

taoyd@sustech.edu.cn

Outline

- ▶ Java's primitive types (基本数据类型)
- ▶ Arithmetic computation (算术运算)
- ▶ Evaluation order of arithmetic expressions (求值顺序)



Data types

- ▶ All programs are composed of **data** and **operations** on the data.
- ▶ A **data type** tells the computer how the programmer intends to use the data
 - What is the meaning of the data?
 - What operations can be done on the data?
 - How to store the data in memory? *占空间不同*
基本
- ▶ Java supports different types of data: **primitive data types** (e.g., numbers) and **complex data types** (e.g., Strings, Objects)

复杂



Primitive data types

用以表示

- ▶ Java has eight primitive types (基本数据类型)
 - Integral types (整数类型): `byte`, `short`, `int`, `long`
 - Floating-point types (浮点数类型): `float`, `double`
 - The `boolean` data type (布尔类型)
 - The `char` data type (字符类型)



Integral data types (Integers)

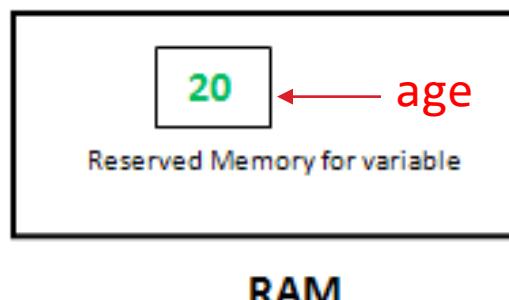
Type	Size	Range
byte	<u>8 bits</u>	-128 to +127
short	16 bits 2^{16}	-32,768 to +32,767
int	32 bits 2^{32}	(about) -2 billion to +2 billion
long	64 bits 2^{64}	(about) -10E18 to +10E18

Example: int age = 20;

Meaning of `int age = 20;`

变量名

- ▶ The statement tells the computer to
 - Allocate space in memory to hold data of `int` type
 - Give the memory location a name “age”, such as we can refer to the data stored in the location using the name in the program (we say we created a **variable** named `age`)
 - Store the value 20 to the allocated space



<https://www.geeksforgeeks.org/variables-in-java/>



Floating-Point Numbers

Type	Size	Range
float (单精度)	32 bits	-3.4E+38 to +3.4E+38
double (双精度)	64 bits	-1.7E+308 to 1.7E+308

默认是 double

Example:

- `double pi = 3.1415926;` 加上 f 才是 float
- `float f = 234.5f;`

告诉类型为
float

The value 234.5 by default is of type double, so f is needed to tell the compiler this is a value of float type



The precision of double and float

- ▶ The **float** type: single-precision floating-point number
 - A float has approximately 7 decimal digits 小数点后
- ▶ The **double** type: double-precision floating-point number
 - A double has approximately 16 decimal digits

```
float f = 1.2345678990922222f; // 16 decimal digits
double d = 1.2222222222222222; // 20 decimal digits
System.out.println("f = " + f + "\t" + "d = " + d);
```

f = 1.2345679

d = 1.222222222222223



Think about this

- ▶ Why computers cannot store real numbers of infinite precisions (such as the irrational number π)? (无理数)
- ▶ It would otherwise require infinite memory (resources are finite in computers). This is why the built-in primitive types can only represent a range of values.

The boolean data type

判断

- ▶ Represents one bit of information (the real size in memory depends on JVM, could be 8 bits)
- ▶ Has only two possible values: `true` and `false`
- ▶ Often used as simple flags for tracking program conditions

Example: `boolean isChecked = true;`

The char data type

* Unicode 表

映射表

- ▶ Represents a single 16-bit **Unicode character**
16进制
- ▶ Ranges from '\u0000' to '\uffff': 65536 characters, covering characters of most modern languages and a large number of symbols

字符

```
char c1 = 'a';
```

String 字符串

'\n____'

韩文字符

不同进制转换

```
char c2 = '\u05357';
```

```
char c3 = '\u79d1';
```

Prints: 南 科 大

```
char c4 = '\u5927';
```

打印 相应位置

```
System.out.printf("%c %c %c", c2, c3, c4);
```

println
→可格式化

Outline

- ▶ Java's primitive types
- ▶ Arithmetic computation (算术运算)
- ▶ Evaluation order of arithmetic expressions



Adding two integers

引入 类 调用输入功能

```
import java.util.Scanner; 定义变量  
public class Addition {  
  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
  
        int number1 = 1; 不执行 提示信息  
        System.out.print("Enter the second integer: ");  
        int number2 = input.nextInt();  
  
        int sum = 0;  
        sum = number1 + number2;  
        System.out.printf("Sum is %d\n", sum);  
    }  
}
```

The screenshot shows a Java code editor with the following code:

```
package lecture1;  
  
import java.util.Scanner;  
  
public class Addition {  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        int number1 = 1;  
  
        System.out.print("Enter the second integer: ");  
        int number2 = input.nextInt();  
  
        int sum = 0;  
        sum = number1 + number2;  
        System.out.printf("Sum is %d\n", sum);  
    }  
}
```

The code uses the `Scanner` class to read input from the console. The variable `number1` is initialized to 1. The program prompts the user to enter the second integer and reads it using `nextInt()`. It then calculates the sum of `number1` and the user input, and prints the result using `printf`.



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;
        System.out.printf("Sum is %d\n", sum);
    }
}
```

- `import` helps the compiler locate a class (e.g., `Scanner`) that is used in this program
- In Java, related classes are grouped into `packages`, e.g., `java.util` package provides commonly-used library classes.
- These classes are collectively called Java class library or Java Application Programming Interface (Java API)



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;
        System.out.printf("Sum is %d\n", sum);
    }
}
```

- ▶ The `Scanner` class enables a program to **read input data**
- ▶ The data can come from different sources, such as the keyboard or a file on disk
- ▶ Standard input object, `System.in`, enables a program to read input data typed by the user



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;
        System.out.printf("Sum is %d\n", sum);
    }
}
```

Variable Declaration

- Variables must be declared with a **type** (e.g., `Scanner`) and a **name** (e.g., `input`) before use
- A variable's **type** specifies what kind of information is stored at that location in memory
- A variable's **name** enables the program to access the value of the variable in memory



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;
        System.out.printf("Sum is %d\n", sum);
    }
}
```

- ▶ The **new** keyword creates an object (we will talk more later)
- ▶ The **assignment operator** = assigns the value on its right to the operand on its left. Here, the input variable will point to the scanner object.



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;
        System.out.printf("Sum is %d\n", sum);
    }
}
```

- ▶ Declare a variable named “number1” with type `int`
- ▶ Assign value 1 to the variable “number1”



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;
        System.out.printf("Sum is %d\n", sum);
    }
}
```

- ▶ Print a message that directs the user to take a specific action
- ▶ Scanner method `nextInt` obtains an integer from the user. The program waits until the user types the number on the keyboard and press the Enter key to submit the number (the method is blocking). 
- ▶ Assign the input value directly to the variable “number2”



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0;
        sum = number1 + number2;

        System.out.printf("Sum is %d\n", sum);
    }
}
```

- ▶ **Expressions:** Portions of statements that contain calculations (e.g., `number1+number2`)
- ▶ The computer **reads** the values of `number1` and `number2` from memory, adds the two values and **stores** the result to the memory location represented by `sum`
- ▶ Note that `sum`'s initial value 0 will be replaced by the new value



Adding two integers

```
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int number1 = 1;

        System.out.print("Enter the second integer: ");
        int number2 = input.nextInt();

        int sum = 0; 声明 + 赋值
        sum = number1 + number2; 变分部分 或 int sum;

        System.out.printf("Sum is %d\n", sum); 拼打印
    }
}
```

int sum = $n_1 + n_2$
或 int sum;
* 只声明整型默认为 0 但运算不可直接
但直接打印错误编译

printf(): Format specifier %d is a placeholder for an int value

The letter 'd' stands for "decimal integer"

Arithmetic operators

Java has five **binary** arithmetic operators (they operate on two operands)

2元运算符

操作符

Operator	Use	Description
+	$op1 + op2$	Adds op1 and op2; also used to concatenate strings
-	$op1 - op2$	Subtracts op2 from op1
*	$op1 * op2$	Multiplies op1 by op2
/	$op1 / op2$	Divides op1 by op2 去尾
%	$op1 \% op2$	Computes the remainder of dividing op1 by op2

Examples

- 整型
- `int x = 3; int y = 2; int z = x / y;` 去尾不四舍五入
- Integer division yields an integer quotient. The fractional part is simply discarded (z gets the value 1)
-
- `int a = 10; int b = 3; int c = a % b;` 取 mod
- c gets the value 1 (the remainder of dividing 10 by 3 is 1)



Evaluation order 优先级

- ▶ An arithmetic expression may contain multiple operators and operands (e.g., $1 + 2 * 5$)
- ▶ The order in which the operators get evaluated depends on their **precedence** (优先级) and **associativity** (结合性)

Precedence of operators

- ▶ Precedence specifies **the priority of an operator**
- ▶ *, / and % operators have the same level of precedence
 - + and – have the same level of precedence
- ▶ *, / and % have higher **>** precedence than + and –
- ▶ So, in expression $1 + 2 * 5$, the multiplication operator will be applied first.

相同优先级：从左到右

Associativity of operators

- ▶ In case there are multiple operators of the same precedence in an expression, their evaluation order is determined by their **associativity**
- ▶ If an expression contains multiple *, / and % operators, they are applied **from the left to right**
- ▶ If an expression contains multiple + and - operators, they are also applied **from the left to right**



Step 1. $y = 2 * 5 * 5 + 3 * 5 + 7;$ (Leftmost multiplication)

2 * 5 is 10

Step 2. $y = 10 * 5 + 3 * 5 + 7;$ (Leftmost multiplication)

10 * 5 is 50

Step 3. $y = 50 + 3 * 5 + 7;$ (Multiplication before addition)

3 * 5 is 15

Step 4. $y = 50 + 15 + 7;$ (Leftmost addition)

50 + 15 is 65

Step 5. $y = 65 + 7;$ (Last addition)

65 + 7 is 72

Step 6. $y = 72$ (Last operation—place 72 in y)



Parentheses in expressions

- ▶ In Java, parentheses operator () has the highest level of precedence
- ▶ In expression $(1 + 2) * 3$, the addition will be done first because of the parentheses
- ▶ Parentheses have left associativity.
- ▶ In expression $(1 + 2) * (3 + 4)$, $1 + 2$ will be done first
- ▶ In case of **nested parentheses**, the expression in the innermost set of parentheses is evaluated first: $((a + b) * c)$



Conditional expressions

条件运算符

- ▶ An expression that can be **true** or **false**
- ▶ Conditional expressions involve two types of operators:
 - Equality operators (相同运算符): **==**, **!=** **写法不同**
 - Relational operators (关系运算符): **>**, **<**, **>=**, **<=**
- ▶ Some refer to these two types as “comparison operators”



二元賦值 不判断相等

Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
<i>Equality operators</i>			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
<i>Relational operators</i>			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	≥	x ≥ y	x is greater than or equal to y
≤	≤	x ≤ y	x is less than or equal to y

Precedence and associativity

Operators	Associativity	Type
*	left to right	multiplicative
/		
%		
+	left to right	additive
-		
<	left to right	relational
<=		
>	left to right	
>=		
==	left to right	equality
!=		
① 最低	right to left	assignment

boolean isSame = $1+3 \text{ } != 5*3;$
 ② ③ ④
 true



Displaying formatted data with printf

```
public class Printer {  
    public static void main(String[] args) {  
        System.out.printf("%d\n", -123);  
        System.out.printf("Total: $%.2f\n", 1234.5678);  
        System.out.printf("%s World", "Hello");  
    }  
}
```

Printer prints the following text on the console:

-123

Total: \$1234.57

Hello World

打印有格式信息

- `System.out.printf()` method takes a **format string** (格式字符串) as an argument.
- The **format specifiers** (格式说明符) begin with a percent sign (%) and are followed by a character that represents the data type (e.g., `%s` is a placeholder for a string)

Tabulating output with printf

radius	perimeter	area
1	6.2832	3.1416
2	12.5664	12.5664
3	18.8496	28.2743
4	25.1327	50.2655



How to generate beautiful tables using printf?



Here is the magic code

```
double pi = Math.PI; 左对齐
System.out.printf("%-20d%-20s%-20s\n", "radius", "perimeter", "area");
System.out.printf("%-20d%-20.4f%-20.4f\n", 1, 2*pi*1, pi*1*1);
System.out.printf("%-20d%-20.4f%-20.4f\n", 2, 2*pi*2, pi*2*2);
System.out.printf("%-20d%-20.4f%-20.4f\n", 3, 2*pi*3, pi*3*3);
System.out.printf("%-20d%-20.4f%-20.4f\n", 4, 2*pi*4, pi*4*4);
```

Please decode the format strings by yourself.

Or visit the link below.

https://www.cs.colostate.edu/~cs160/Summer16/resources/Java_printf_method_quick_reference.pdf



Appendix – Terms

- ▶ Comment 注释 End-of-line comments 行末注释 Syntax error 语法错误
- ▶ String 字符串 Command window 命令窗口 Argument 参数 Cursor 光标
- ▶ Console 控制台 White-space characters 空白字符 Escape character 转义字符
- ▶ Carriage return 回车 Format string 格式字符串 Format specifier 格式说明符
- ▶ Primitive types 基本数据类型 Floating-point number 浮点数
- ▶ Decimal digits 小数位数 Unicode 万国码
- ▶ Standard input/output 标准输入输出 Assignment operator 赋值运算/操作符
- ▶ Prompt 提示符 Binary arithmetic operator 二元算术操作符
- ▶ Precedence 优先级 Associativity 结合性 Nested parentheses 嵌套的圆括号
- ▶ Equality operator 相同运算符 Relational operator 关系运算符