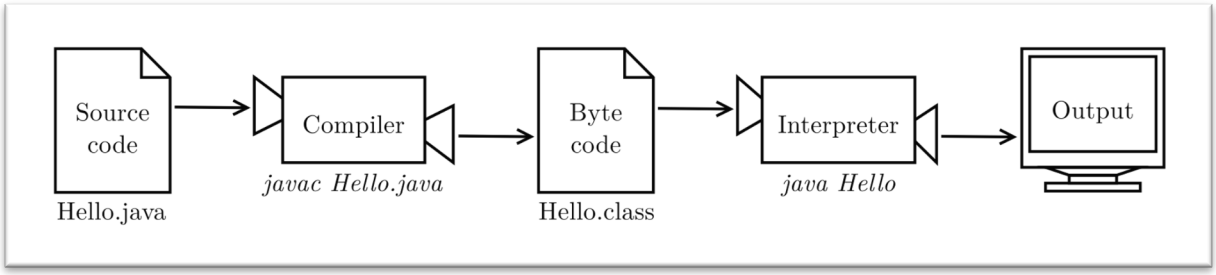
AP Computer Science A

Lecture note 01

Student: _____ Instructor: 杜博识 Dubos Date: _____

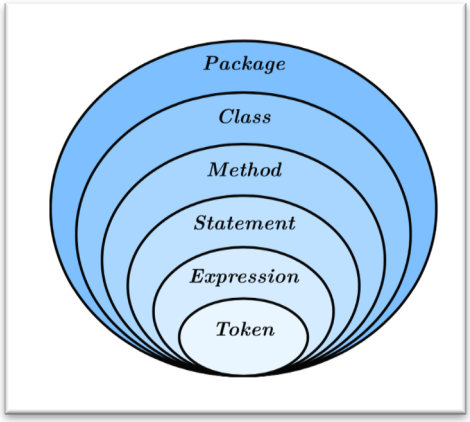
Computer science is the science of algorithms, including their discovery and analysis. An **algorithm** is a sequence of steps that specifies how to solve a problem.

The process of compiling and running a Java program



Source code 源代码	the program written in high-level language such as Java, Python, and C++
Compiler 编译器	it reads the entire program source code and translate it into byte code completely before the program starts running
Byte code 字节代码	the low-level language (“machine language”) that is executable for the computer
Interpreter 解释器	it processes the program a little at a time, alternatively reading lines and performing computations

Java Program Structure



Package 包	A package is a collection of related classes. • AP CS A exam does not require knowledge of packages. You will not be expected to write any `import` statements.	
Class 类	A class is a collection of methods • By convention, class names begin with a capital letter	
Method 方法	A method is a named sequence of statements	
Statement 语句	A statement is a line of code that performs a basic operation	
Expression 表达式	An expression represents a **SINGLE VALUE** to be computed • When the program runs, each variable is replaced by its current value	
Token 符号	Expressions are made up of tokens, which are the basic elements of a program, including numbers, variable names, operators, keywords, and punctuation like parentheses, braces, and semicolons.	
Flow of execution	• Execution always begins at the first statement of `main`, regardless of where it is in the source code • Statements are executed one at a time, in order, until you reach a method invocation. Then you jump to the first line of the invoked method, execute all the statements there, and then come back and pick up exactly where you left off.	

Declaring variables

Variable 变量	A variable is a named location that stores a value	
Declaration 声明	Declaring variables with names and types • When you declare a variable, you create a named storage location • When a variable name contains more than one word, it is conventional to capitalize the first letter of each word except the first • Variable names are case-sensitive	
Java Keywords Java 关键字	See the list below. • There are about 50 Java keywords that are used by the compiler to analyze the structure of the program. They are not allowed to be used as variable names	
Type 类型	The type of a variable determines what kind of value the variable can store	
Primitive types 基本类型	In AP CS A exam, we are expected to know: • `int`: an integer. • `double`: a double precision floating-point number	

	• `boolean`: just two values, `true` or `false`.	
Storage of number	• Integer values in Java are stored as a string of bits (binary digits) • `int` in Java uses four bytes (32 bits). Taking one bit as +/- sign, the largest positive integer stored is 2^{31}-1 and the smallest negative is -2^{31} • the floating-point number is stored as: sign * mantissa * $2^{exponent}$ • `double` in Java uses eight bytes (64 bits). Taking one bit as +/- sign, 52 bits as **mantissa** 尾数, and 11 bits as the exponent. • Because floating-point numbers are converted to binary, most cannot be represented exactly, leading to **round-off error** 舍入误差	
Type cast 类型转换	• `double pi = 3.14;` `int x = (int) pi;` coverts the double variable `pi` into an integer 3 and assign to `x` • Type cast takes precedence over arithmetic operations, so `int y = (int)pi * 20;` the value of `y` should be 3 * 20 = 60, not (int)(3.14 * 20) = 62	
Assignment 赋值	Assignment the right-hand-side of = (which is a value) to the left-hand-side of = (which is a variable name and refer to a storage location) • When you assign a value to a variable, you update its value • The variable should have the same type as the value you assign to it	
Initialization 初始化	Assignment of a variable for the first time • Variables must be initialized before they are used	
`final` **variables**	A `final` variable is used to name a quantity whose value will not change. • By convention the name of `final` variables are capitalized • A `final` variable can be declared without initializing it immediately • `final` variables will not be reassigned.	

Assignment Operators		
Operator	Meaning	Example
`=`	Assigned as	`n = 5;` means n is assigned 5
`+=`	Increase by	`n += 5;` means `n = n+5;`
`-=`	Decrease by	`n -= a;` means `n = n-a;`
`*=`	Multiplied by	`n *= a;` means `n = n*a;`
`/=`	Divided by	`n /= a;` means `n = n/a;`
`%=`		`n %= a;` means `n = n%a;`

Name	*Range*	*Storage Size*	
`byte`	-2^7 to $2^7 - 1$ (−128 to 127)	8-bit signed	`byte` type
`short`	-2^{15} to $2^{15} - 1$ (−32768 to 32767)	16-bit signed	`short` type
`int`	-2^{31} to $2^{31} - 1$ (−2147483648 to 2147483647)	32-bit signed	`int` type
`long`	-2^{63} to $2^{63} - 1$ (i.e., −9223372036854775808 to 9223372036854775807)	64-bit signed	`long` type
`float`	Negative range: −3.4028235E + 38 to −1.4E − 45 Positive range: 1.4E − 45 to 3.4028235E + 38	32-bit IEEE 754	`float` type
`double`	Negative range: −1.7976931348623157E + 308 to −4.9E − 324 Positive range: 4.9E − 324 to 1.7976931348623157E + 308	64-bit IEEE 754	`double` type

Java Language Keywords

Here is a list of keywords in the Java programming language. You cannot use any of the following as identifiers in your programs. The keywords `const` and `goto` are reserved, even though they are not currently used. `true`, `false`, and `null` might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Arithmetic operators

Arithmetic operators 运算符	Addition `+`, subtraction `-`, multiplication `*`, division `/`, modulus `%` • **Integer division**: both dividend and divisor are of `int` type • Integer division always rounds toward zero • How to design an “Smart Integer Division” so that the result rounds toward the close integer? • **Floating-points division**: dividend or divisor or both are `double`	

String and escape sequence

String 字符串	Phrases that appears in quotation marks	
String	For Strings, the + operator joins Strings before and after it end	`1 + 2 + "String"`

concatenation 字符串串连	to end.	`"String" + 1 + 2`

Escape sequence
转义字符
	A sequence of characters that represents a special character	`\n`
		`\t`
		`\”`
		`\\`

Formatting output	`System.out.printf();`	

Types of errors

Compile-time error	When you violate the syntax rules of the Java language.	
Run-time error	The program will compile without errors. Run-time errors occur while the interpreter is executing byte code. Also called “exceptions”. • `ArithmeticException` • `IllegalArgumentException` • `NullPointerException` • `ClassCastException` • `ArrayIndexOutOfBoundsException` • `IndexOutOfBoundsException`	
Logic error	The program will compile and run without generating error messages, but it will not do the right thing.	

Void method vs. value method

Parameter 参数	The parentheses after the method name contains a list of variables called parameters. • When you write a method, you name the parameters and specify the type of each variable separately.	
Argument 自变量值	When you use a method, you provide the arguments to the parameters. • Before the method executes, the arguments get assigned to the parameters. This process is called **parameter passing** 参数传递. • An argument can be any kind of expression. • The value you provide as argument must have the same type as the parameter. The parameter list indicates what arguments are required. • Parameters are declared and assigned in their own	

	methods, so they only exist inside their own methods. The same to other **local variables** 局部变量 which are defined in a certain method (rather than in the class).	

Binary, Octal and Hexadecimal Numbers

Binary 二进制	Leading with 0B `System.out.println(0B1111);`	
Octal 八进制	Leading with 0 `System.out.println(0B17);`	
Hexadecimal 十六进制	Leading with 0X `System.out.println(0XF);` • Any hexadecimal expands to four bits • To convert a binary number to hex, convert in groups of four digits from right to left	

Source code of LN01: https://github.com/Duboshi/AP_CS_A/blob/main/src/LN01.java

```java
/*
 * Lecture note 01
 * Introduction to Java Programming
 *
 * Block comment段落注释
 */

//line comment行内注释

import java.util.Scanner;                          //Step 1 of input

public class LN01 {//class类
    private static Scanner in;                     //Step 2 of input

    public static void main(String[] args) {       //method方法（函数）
        System.out.println("Hello World!");            //statement语句

        int n;
        //Declaration声明/建立 of an int (type) variables named n
        n = 5 + 3;                                 //Assignment赋值
        System.out.println(n);
        n = n * 4;                                 //Assignment
        System.out.println(n);

        double x = 5.5;
        double y = 2.2;
        System.out.println(x + y);

        //Primitive types: double浮点数/实数/小数, int整型变量（整数）
        System.out.println(x + y); //+, -, *, /
        n = 8;
        int m = 5;

        //Arithmetic operators: + - * / %
        System.out.println(8/5);       //integer division
        System.out.println(8/5.0);
        System.out.println(8%5);       //modulus operator, remainder of 8 divided by 5
        System.out.println(8%(-5));    //3
        System.out.println((-8)%5);    //-3, same sign as the dividend, not as divisor
        System.out.println(x%y);               //round-off error

        System.out.println(0.1 + 0.1);
        System.out.println(0.1 + 0.1 + 0.1);   //round-off error

```

```
        //Assignment operator
        n = 6;
        n *= 3;
        System.out.println(n); //18. n*=3 means n = n*3.
        n %= 5;
        System.out.println(n);     //3

        //String字符串
        String s1 = "Ai", s2 = "Zeng";
        System.out.println(s1.compareTo(s2));
        System.out.println(s1 + s2);                        //concatenation of strings
        System.out.println(1 + 2 + "String");               //3String
        System.out.println("String" + 1 + 2);               //String12

        //Escape sequence转义字符 and formatted print格式输出
        System.out.print("Hello, Senior 3~\n");             //escape sequence \n, backslash
        System.out.println("He said \"Hello, World!\"");    // escape sequence\"
        System.out.println("\\^0^/");                       // escape sequence\\

        System.out.println("Chapter 01");
        System.out.println("\tSection a");
        System.out.println("\t\tParagraph");                // escape sequence\t
        System.out.println("\tSection b");

        int hour = 11, min = 10, snd = 7;
        System.out.println(hour + ":" + min + ":" + snd);
        System.out.printf("%02d:%02d:%02d\n", hour, min, snd);
        System.out.printf("%2d:%2d:%2d\n", hour, min, snd);
        //printf method: %d, %2d, %02d, %f, %2f, %02f

        //0x: octal八进制, 1 2 3 4 5 6 7 10 11
        double percent;
        int sum = 3600*hour + 60*min + snd;
        System.out.println(sum);
        percent = sum/86400.0;
        System.out.println(percent);
        //Compile-time error, run-time error, logic error
        //System.out.println(5/0);

        //input and Scanner.in
        in = new Scanner(System.in);                //Step 3 of input
        System.out.println("What is your name?");
        String name = in.nextLine();                //Step 4 of input
        System.out.println("Hello, " + name);
```

```
89          System.out.println("What is your height in cm?");
90          int h = in.nextInt();
91          final double cmPerInch = 2.54;              //constant
92          final int inchPerFoot = 12;
93          int feet = (int)(h/cmPerInch/12);           //type cast
94          int inch = (int)(h/cmPerInch%12);
95          System.out.println(feet + " feet " + inch + " inches.");
96
97          //void method vs. value method
98          System.out.println(sum(2, 5));
99          voidsum(2, 5);
100         System.out.println(sum(1.1, 2.2));
101
102         //Recursion
103         countdown(5);
104     }
105
106     public static int sum(int a, int b) {          //value method which returns an int
107         return a + b;
108     }
109
110     public static void voidsum(int a, int b) {     //void method
111         System.out.println(a + b);
112     }
113
114     public static double sum(double a, double b) {  //value method which returns a double
115         return a + b;
116     }
117
118     public static void countdown(int n) {
119         if (n>0) {
120             System.out.println(n);
121             countdown(n-1);                        //recursion递归
122         } else
123             System.out.println("GO!!!");
124     }
125 }
```