Primitive Types & Using Objects

AP Computer Science A ▪ Unit 1

AP Computer Science A

```
int x = 5;
```

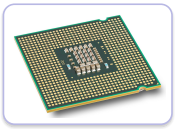

What we will cover

1. Programs & compilers
2. Variables & data types
3. Expressions & casting
4. Methods & the Math class
5. Objects & strings

1 Foundations

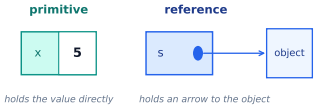
From source code to running program

An **algorithm** 算法 is a step-by-step plan; a **program** 程序 is that plan in a **programming language** 编程语言 like Java. The compiler turns human-readable source into **bytecode** the JVM runs.



Cpu chips

Variables & primitive types



A **variable** 变量 is a named box that stores a value of a fixed **type** 类型.

> Java's main **primitive types** 基本类型 are 'int' (whole numbers), 'double' (decimals), and 'boolean' ('true'/'false').

Primitive Types & Using Objects
└Foundations

Variables & primitive types

type	*stores*	*example*
`int`	whole number	`7`
`double`	decimal number	`3.5`
`char`	one character	`'A'`
`String`	text (a sequence of	`"hello"`
`boolean`	true or false	`true`

A **variable** 变量 is a named box that stores a value of a fixed **type** 类型.

> Java's main **primitive types** 基本类型 are 'int' (whole numbers), 'double' (decimals), and 'boolean' ('true'/'false').

2 Expressions

Expressions, casting & range

An **expression** 表达式 combines values with **operators** 运算符 (+ − ∗ / %), following order of operations.

Integer division & casting

`7 / 2` is 3 (integer division drops the remainder); `7 % 2` is 1.

Cast 类型转换 to get a decimal: `(double) 7 / 2` is `3.5`. A value too big for its type causes **overflow** 溢出.

Compound assignment 复合赋值: `x += 3` means `x = x + 3`.

3 Methods & Math

Methods, the API & the Math class

A **method** 方法 is named, reusable code. Its **signature** 方法签名 is its name, parameters & return type.

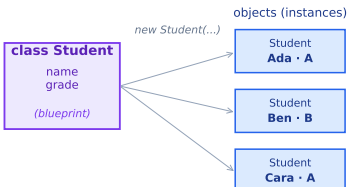
API 应用程序接口 & libraries 库
documentation of ready-made classes to reuse

`Math` class
`Math.sqrt(9)`, `Math.pow`, `Math.random()`

A **comment** 注释 (`// ...`) documents code for humans; the compiler ignores it.

4 Objects & strings

Objects: instances of classes

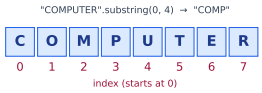


A **class** 类 is a blueprint; an **object** 对象 is a built **instance** 实例.

Instantiate 实例化 with `new`: `Scanner s = new Scanner(...);`

Call **instance methods** 实例方法 on an object with the dot operator: `s.nextInt()`.

String manipulation



A **String** 字符串 is an object – an ordered sequence of characters, indexed from **0**.

- `s.length()`
- `s.substring(0, 4)`
- `s.indexOf("x")`

Concatenate 拼接 with `+`: `"Hi, " + name`.

Key points

Java **source** → compiler → **bytecode** → JVM

Strongly typed: `int`, `double`, `boolean`

`7/2=3`; **cast** for decimals; `x += 3`

A **class** is a blueprint; an **object** is an instance

Worked example – the excluded endpoint

`String s = "COMPUTER";` (indices 0–7)

`s.length()` → 8
`s.substring(0, 4)` → "COMP" (index 4 is excluded)
`s.substring(4)` → "UTER" (index 4 to the end)
`s.indexOf("PU")` → 3
`s.indexOf("X")` → -1 (not found)

Miscounting `substring`'s excluded endpoint is the single most common slip on this paper.

Check yourself

1. `"COMPUTER".substring(0,4)` gives what?
2. What does `indexOf` return when the string is absent?
3. Is `int` a primitive or a reference type?
4. What does the compiler turn source code into?

Answers **1.** `"COMP"` **2.** `-1` **3.** primitive **4.** bytecode, run by the JVM