The Java Language

The Vocabulary & Syntax

Jayashree Nair
Department of Computer Science
Amrita School of Engineering
Amrita Vishwa Vidyapeetham
Amritapuri Campus

jayashree@am.amrita.edu

Overview

- **The Java Keywords** & **Symbols** : **int, public, class**….. & **(), {}, ; ., ….**
- **Statement** : ends with ;
- **Comment** : **//** & **/* */**
- Identifier : no keywords, _ $, starts with alphabet: **x4**, **4x**, **_n_1**, **%y**
- **Variable and DataType** : A Strongly-typed language.
- **Primitive** type and **Reference** Type
  - **Primitive type** : int, float, char, boolean ……
  - **Reference type** : String and Scanner classes
- **String** class – to store sequence of characters - “Amrita”
- **System.out.println()** : The output tool
- **Scanner** class : The input tool.

The Java Keywords

and, between : English

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

Table 2-1 Java Keywords

Separators or Symbols

Symbol	Name	Purpose
( )	Parentheses	Used to contain lists of parameters in method definition and invocation. Also used for defining precedence in expressions, containing expressions in control statements, and surrounding cast types.
{ }	Braces	Used to contain the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes.
[ ]	Brackets	Used to declare array types. Also used when dereferencing array values.
;	Semicolon	Terminates statements.
,	Comma	Separates consecutive identifiers in a variable declaration. Also used to chain statements together inside a **for** statement.
.	Period	Used to separate package names from subpackages and classes. Also used to separate a variable or method from a reference variable.
::	Colons	Used to create a method or constructor reference. (Added by JDK 8.)

Statement in Java

- Words and symbols put together to form statement in Java.
- A statement is a unit of code that does one task or computation.
- Every statement must end with a ;
  - **sum = x + y;**
  - **System.out.println(“Welcome to Java!!!”);**
- **Statements are group inside a block { }**

Comments

- Statements for **documentation** purpose – for **programmers**.
- The contents of a comment are **ignored** by the **compiler**.
- **//** - single line comment
- **/* */** - multiline comment
- **/** */** - documentation comment

```
sum = x + y; // adds two numbers x and y to sum
```

Identifiers

- Identifiers are used to **name things**, such as classes, variables, and methods.
- **Keywords can't** be used.
- Can contain
  - uppercase and lowercase letters a-zA-Z, e.g: - **aNumber**
  - Numbers 0-9, or e.g. – **x2**, **x3**
  - the underscore and e.g. - **_name3**
  - dollar-sign characters. e.g. - **$age**
- Rules:-
  - must **not begin with a number or special characters** except (_ and $) e.g- 2x, 3x , #p

A Sample Code

MyFirstApp.java

```
1  public class MyFirstApp
2  {
3      //This is my first program
4      public static void main(String[] args)
5      {
6          System.out.println("Welcome to the world of Java!!!");
7          /* the above statement will print
8             the string in the console
9           */
10     }
11 }
```

Storage Containers : the Variables

Variables

- Basic **unit of storage** in a Java program.
- A variable is just like a holder – a container – that holds data.
- Variables must have a **type** e.g. **int**, **double**, **String**, **Date**, **boolean**.
- Variables must have a **name** e.g. **studentName**, **studentAge**, **patientAddress,** .

Java is case-sensitive age is different from Age

The primitive data types

numeric (all are signed)

integer

byte	8 bits	-128 to 127
short	16 bits	-32768 to 32767
int	32 bits	-2147483648 to 2147483647
long	64 bits	-huge to huge

floating point

float	32 bits	varies
double	64 bits	varies

boolean and char

boolean	(JVM-specific)	***true*** or ***false***
char	16 bits	0 to 65535

Primitive DataTypes in Java

Table 1 Primitive Types		
Type	Description	Size
`int`	The integer type, with range –2,147,483,648 (`Integer.MIN_VALUE`) . . . 2,147,483,647 (`Integer.MAX_VALUE`, about 2.14 billion)	4 bytes
`byte`	The type describing a single byte, with range –128 . . . 127	1 byte
`short`	The short integer type, with range –32,768 . . . 32,767	2 bytes
`long`	The long integer type, with range –9,223,372,036,854,775,808 . . . 9,223,372,036,854,775,807	8 bytes
`double`	The double-precision floating-point type, with a range of about $\pm 10^{308}$ and about 15 significant decimal digits	8 bytes
`float`	The single-precision floating-point type, with a range of about $\pm 10^{38}$ and about 7 significant decimal digits	4 bytes
`char`	The character type, representing code units in the Unicode encoding scheme (see Computing & Society 4.2 on page 161)	2 bytes
`boolean`	The type with the two truth values `false` and `true` (see Chapter 5)	1 bit

Java Is a Strongly Typed Language

- Every variable must have a **declared type**.
- There are **eight primitive** types in Java.
  - Four are **integer** types – 45, 69096096098, 9
  - Two are **floating-point** number types – 4.3333334, 3.14,
  - One is the **character** type char - ‘A’, ‘c’, ‘9’, ‘u’
  - One is a **boolean** type for truth values – true & false

The Integer types

Table 3.1 Java Integer Types

Type	Storage Requirement	Range (Inclusive)
`int`	4 bytes	–2,147,483,648 to 2,147,483, 647 (just over 2 billion)
`short`	2 bytes	–32,768 to 32,767
`long`	8 bytes	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`byte`	1 byte	–128 to 127

The floating types

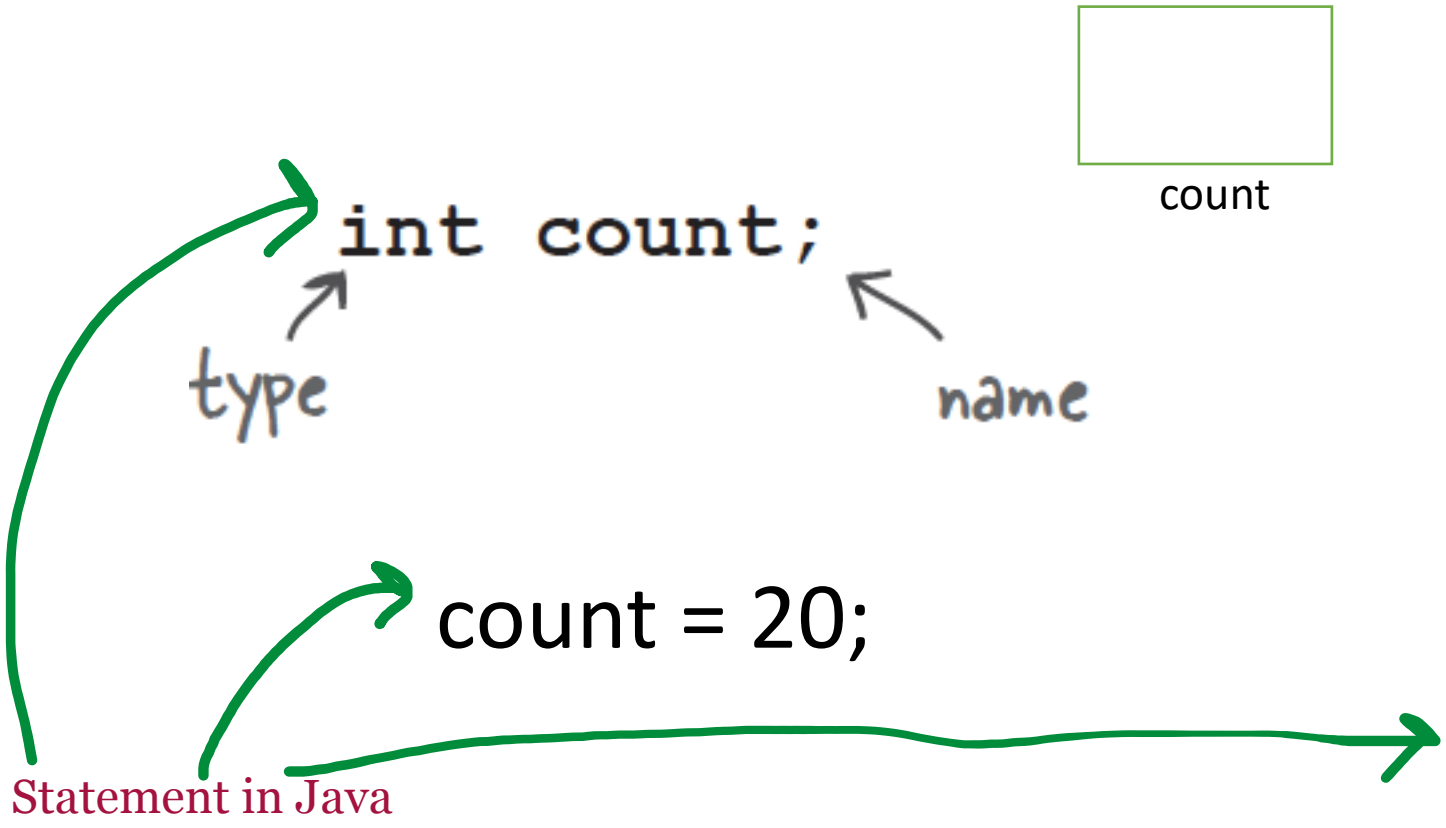
Table 3.2 Floating-Point Types

Type	Storage Requirement	Range
`float`	4 bytes	Approximately ±3.40282347E+38F (6–7 significant decimal digits)
`double`	8 bytes	Approximately ±1.79769313486231570E+308 (15 significant decimal digits)

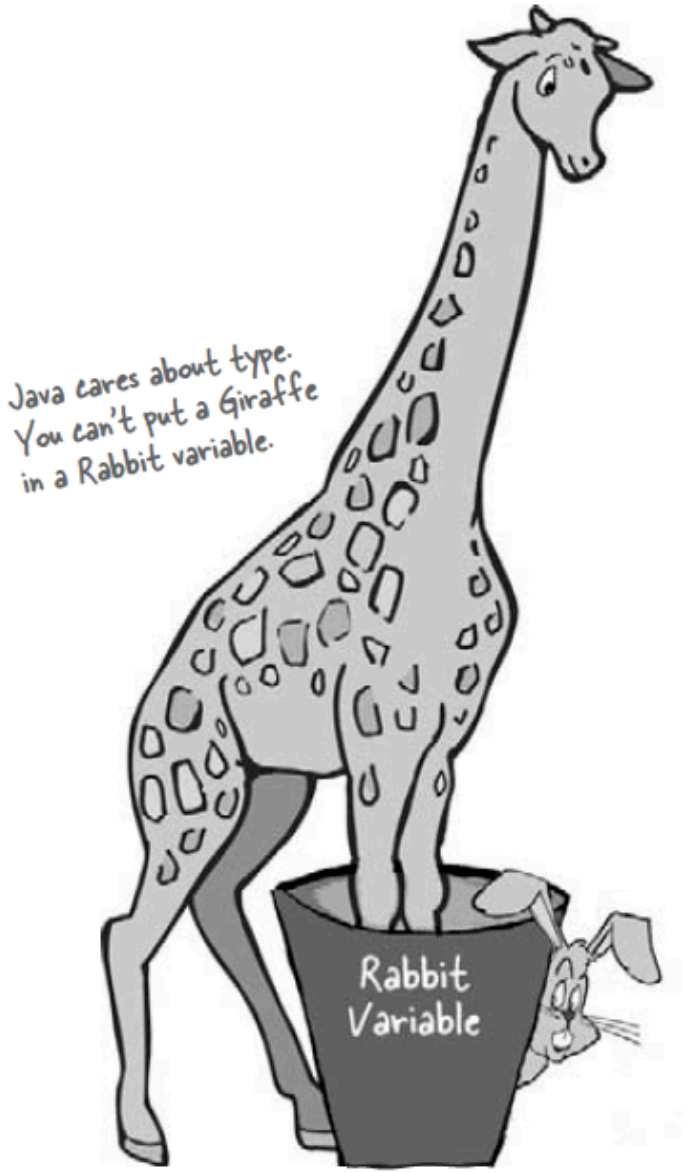
Characters

- Anything inside single quote ‘a’, ‘b’, ‘+’, ‘0’
- The data type used to store characters is char.
- Java uses **Unicode** to represent characters.
- Java char is a 16-bit type.
- The range of a char is 0 to 65,536

Declaring & Initializing a variable to store value




```
int x;
x = 234;
byte b = 89;
boolean isFun = true;
double d = 3456.98;
char c = 'f';
int z = x;
boolean isPunkRock;
isPunkRock = false;
boolean powerOn;
powerOn = isFun;
long big = 3456789;
```



Java cares about types

```
int count;
```

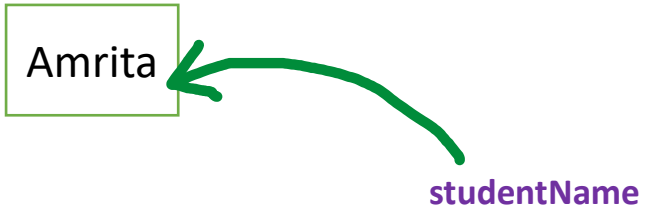
type

name

✗ count = "Amrita"

Reference Types

- Special type – which **do not store values** but **stores the reference** of the value.
- In Java, these are **arrays** and **class objects**.
- E.g. **String**.



The String (type????? Or Class)

- How can you store your name?
- Your Address?
- “Amrita”
- “Amrita School of Engineering”

Strings

- Java strings are **sequences of Unicode characters**
- Java does not have a built-in or primitive string type.
- Instead, the **standard Java library** contains a predefined class called String

String name = “Amrita”;

String address = “Amritapuri Campus”;

Declaring Strings

- String name;
- name = “Amrita”;
- String address = “Amrita Vishwa Vidyapeetham”;
- String phone = “+91-9999912345”

The Output Tool of Java

- System.***out***.println("**Amrita Vishwa Vidyapeetham**");
- System.***out***.println("**name**");
- System.***out***.println("**Your name is**" + **name**);
- System.***out***.println("**Your age is**" + **age**);
- System.***out***.println("**Your address is**" + **address**);

AddTwoNumbers.java

```
public class AddTwoNumbers
{
        public static void main(String[] args)
        {
                int x = 4;
                int y = 5;
                int sum = x + y;
                System.out.println("The sum is :"+sum);
        }
}
```

The input tool of Java

- **Scanner** class helps you to read values from the keyboard.
- Its available in the following **library**:
  - java.util
- You need to **import the library** before using it.

Using the Scanner

- `import java.util.Scanner;`
- ....
- `Scanner in = new Scanner(System.in);`
- `int age = in.nextInt();`
- `double marks = in.nextDouble();`
- `String name = in.next();`
- `String address = in.nextLine();`

AddTwoNumbers.java

```
public class AddTwoNumbers
{
        public static void main(String[] args)
        {
                int x = 4;
                int y = 5;
                int sum = x + y;
                System.out.println("The sum is :"+sum);
        }
}
```

AddTwoNumbers2.java

```java
import java.util.Scanner;

public class AddTwoNumbers2
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter two numbers:");
        int x = in.nextInt();
        int y = in.nextInt();
        int sum = x + y;
        System.out.println("The sum is :"+sum);
        in.close();
    }

}

```

```
<terminated> AddTwoNumbers2 [Java Application]
Enter two numbers:
15 25
The sum is :40
```

Recap

- **The Java Keywords** & **Symbols** : **int, public, class**….. & **(), {}, ; ., ….**
- **Statement** : ends with ; → sum= x+y;
- **Comment** : **//** & **/* */** → sum= x+y; **// add two numbers**
- Identifier : no keywords, _ $, starts with alphabet: **x4**, **4x**, **_n_1**, **%y**
- **Variable and DataType** : A Strongly-typed language.
- **Primitive** type and **Reference** Type
  - **Primitive type** : int, float, char, boolean ……
  - **Reference type** : String and Scanner classes
- **String** class – to store sequence of characters - “Amrita”
- **System.out.println()** : The output tool
- **Scanner** class : The input tool.