# Introduction to Java Basics

Programming in Java

# Contents

# Template of a simple Java program

```
//*************************************************
// comments on the program (authors, purpose, …)
//*************************************************

public class SomeIdentifier
{
   //-----------------------------------------------
   //  comments on the main method
   //-----------------------------------------------
   public static void main (String[] args)
   {
      // declarations of variables and constants
      // statements of the main method
   }
}
```

SomeIdentifier.java

# 1- Comments

- Also called ***inline documentation***
- It is used to explain the purpose of the program and describe processing steps (the algorithm)
- Does not affect how a program works (are ignored by the compiler)
- Can take three forms:

```
// this comment runs to the end of the line
```

```
/*  this comment runs to the terminating
    symbol, even across line breaks
*/
```

```
/** this is a javadoc comment   */
```

# 2- Identifiers

- They are names a programmer gives to variables, classes, methods, etc.
- Rules to create an identifier:
  - Can be made up of:
    - Letters
    - Digits
    - The underscore character (_)
    - And the dollar sign ($)
- Cannot begin with a digit
- Cannot be a ***reserved*** word
- No limit on length

# 2- Identifiers: Java reserved words

- Here is a list of keywords in the Java programming language
- You cannot use any of the following as identifiers in your program
- 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.

# 2- Identifiers: Java reserved words

| | | | | |
|---|---|---|---|---|
| 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

# 2- Identifiers: valid identifier?

- Which of the following is **not** a valid identifier?

  A) abc

  B) ABC

  C) Abc

  D) aBc

  E) a bc

# 2- Identifiers: valid identifier?

- Which of the following is **not** a valid identifier?

  A) abc

  B) ABC

  C) Abc

  D) aBc

  E) a bc

# 2- Identifiers: valid identifier?

- Which of the following is **not** a valid identifier?

  A) a_bc

  B) A$BC

  C) _Abc

  D) 1AbC

  E) $abc

# 2- Identifiers: valid identifier?

- Which of the following is **not** a valid identifier?

  A) a_bc

  B) A$BC

  C) _Abc

  D) 1AbC

  E) $abc

# 2- Identifiers: Guidelines

- Give a significant name!
- Avoid “$”
- By convention:
  - Class names => use camel case
    - Ex: **MyClass**, **Lincoln**
  - Constants => use uppercase, separate with underscore
    - Ex: **MAXIMUM**, **SIZE_LIMIT**
  - Variables, methods, … => start with lowercase
    - Ex: **myVariable**, **lowestGradeOfClass**

# 2- Identifiers: Guidelines

- Avoid predefined identifiers:
  - Although they can be redefined, it is confusing and dangerous
    - Ex: **System**, **String**, **println**, ...

- Remember: Java is ***case sensitive***

# 2- Identifiers: examples

| Identifier | Correct or not? |
|---|---|
| `GST` | |
| `PriceBeforeTax` | |
| `Student_3` | |
| `student#3` | |
| `Shipping&HandlingFee` | |
| `Class` | |
| `__123` | |
| `the account` | |
| `1floor` | |

# 2- Identifiers: examples

| Identifier | Correct or not? |
|---|---|
| `GST` | YES |
| `PriceBeforeTax` | YES |
| `Student_3` | YES |
| `student#3` | NO |
| `Shipping&HandlingFee` | NO |
| `Class` | YES |
| `__123` | YES |
| `the account` | NO |
| `1floor` | NO |

# 3- Indentation

- Spaces, blank lines and tabs are called ***white spaces***
- White space is used to separate words and symbols in a program
- Extra white spaces are ignored
- Programs should be formatted to enhance readability, using consistent indentation

# 3- Indentation: bad indentation example 1

```
//*************************************************************
//  Lincoln2.java
//  Demonstrates a poorly formatted, though valid,
//  Program.
//*************************************************************

public class Lincoln2{public static void
   main(String[]args){
System.out.println("A quote by Abraham Lincoln:");
System.out.println("Whatever you are, be a good one.");}}
```

# 3- Indentation: bad indentation example 2

```
//*************************************************************************
//  Lincoln3.java
//  Demonstrates another valid program that is poorly formatted.
//*************************************************************************
          public       class
         Lincoln3
       {
                  public
       static
            void
      main
            (
   String
                []
        args
                           )
  {
      System.out.println        (
"A quote by Abraham Lincoln:"          )
      ;         System.out.println
                 (
           "Whatever you are, be a good one."
          )
      ;
   }
              }
```

# 3- Indentation: good indentation example

```
//********************************************************
//  Lincoln3.java
//  Demonstrates a properly formatted program.
//********************************************************

public class Lincoln3
{
   public static void main(String[]args)
   {
        System.out.println("A quote by Abraham Lincoln:");
        System.out.println("Whatever you are, be a good one.");
   }
}
```

# 4- Primitive Types: 8 primitive data types in Java

- Numeric
  - Four types to represent integers (ex: 3, -5)

    **byte**, **short**, **int** and **long**
  - Two types to represent floating point numbers (ex: 3.5)

    **float** and **double**
- Characters (ex: ‘a’)

  **char**
- Boolean values (true or false)

  **boolean**

# 4- Primitive Types: Numerical Types

- The difference between:
  - **byte**, **short**, **int, long** AND **float, double** is their size (so the values they can store)

Display 1.2 **Primitive Types**

| TYPE NAME | KIND OF VALUE | MEMORY USED | SIZE RANGE |
|---|---|---|---|
| boolean | true or false | 1 byte | not applicable |
| char | single character (Unicode) | 2 bytes | all Unicode characters |
| byte | integer | 1 byte | −128 to 127 |
| short | integer | 2 bytes | −32768 to 32767 |
| int | integer | 4 bytes | −2147483648 to 2147483647 |
| long | integer | 8 bytes | −9223372036854775808 to 9223372036854775807 |

# 4- Primitive Types: Numerical Types

- The difference between:
  - **byte**, **short**, **int, long** AND **float, double** is their size (so the values they can store)

Display 1.2 **Primitive Types**

| TYPE NAME | KIND OF VALUE | MEMORY USED | SIZE RANGE |
|---|---|---|---|
| `float` | floating-point number | 4 bytes | $-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$ |
| `double` | floating-point number | 8 bytes | $\pm 1.76769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$ |

# 4- Primitive Types: Round-off errors in floating-point numbers

- Floating point numbers are only approximate quantities
  - Mathematically, the floating-point number 1.0/3.0 is equal to 0.333333333…
  - A computer may store 1.0/3.0 as something like 0.3333333333

# 4- Primitive Types: Characters

- A **char** stores a **single** character
- Delimited by **single** quotes:
  - ‘a’ ‘X’ ‘7’ ‘$’ ‘ , ‘ ‘\n’
- characters are ordered according to a ***character*** set
- each character corresponds to a unique number code
- Java uses the Unicode character set
  - 16 bits per character, so 65,536 possible characters
  - Unicode is an international character set, containing symbols and characters from languages with different alphabets

# 4- Primitive Types: Characters

- The ASCII character set is older and smaller than Unicode, but is still popular.

- The ASCII characters are a subset of the Unicode character set.

# 4- Primitive Types: Characters

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | NUL | 26 | 1A | SUB | 52 | 34 | 4 | 78 | 4E | N | 104 | 68 | h |
| 1 | 01 | SOH | 27 | 1B | ESC | 53 | 35 | 5 | 79 | 4F | O | 105 | 69 | i |
| 2 | 02 | STX | 28 | 1C | FS | 54 | 36 | 6 | 80 | 50 | P | 106 | 6A | j |
| 3 | 03 | ETX | 29 | 1D | GS | 55 | 37 | 7 | 81 | 51 | Q | 107 | 6B | k |
| 4 | 04 | EOT | 30 | 1E | RS | 56 | 38 | 8 | 82 | 52 | R | 108 | 6C | l |
| 5 | 05 | ENQ | 31 | 1F | US | 57 | 39 | 9 | 83 | 53 | S | 109 | 6D | m |
| 6 | 06 | ACK | 32 | 20 | space | 58 | 3A | : | 84 | 54 | T | 110 | 6E | n |
| 7 | 07 | BEL | 33 | 21 | ! | 59 | 3B | ; | 85 | 55 | U | 111 | 6F | o |
| 8 | 08 | BS | 34 | 22 | " | 60 | 3C | < | 86 | 56 | V | 112 | 70 | p |
| 9 | 09 | HT | 35 | 23 | # | 61 | 3D | = | 87 | 57 | W | 113 | 71 | q |
| 10 | 0A | LF | 36 | 24 | $ | 62 | 3E | > | 88 | 58 | X | 114 | 72 | r |
| 11 | 0B | VT | 37 | 25 | % | 63 | 3F | ? | 89 | 59 | Y | 115 | 73 | s |
| 12 | 0C | FF | 38 | 26 | & | 64 | 40 | @ | 90 | 5A | Z | 116 | 74 | t |
| 13 | 0D | CR | 39 | 27 | ' | 65 | 41 | A | 91 | 5B | [ | 117 | 75 | u |
| 14 | 0E | SO | 40 | 28 | ( | 66 | 42 | B | 92 | 5C | \ | 118 | 76 | v |
| 15 | 0F | SI | 41 | 29 | ) | 67 | 43 | C | 93 | 5D | ] | 119 | 77 | w |
| 16 | 10 | DLE | 42 | 2A | * | 68 | 44 | D | 94 | 5E | ^ | 120 | 78 | x |
| 17 | 11 | DC1 | 43 | 2B | + | 69 | 45 | E | 95 | 5F | _ | 121 | 79 | y |
| 18 | 12 | DC2 | 44 | 2C | , | 70 | 46 | F | 96 | 60 | ` | 122 | 7A | z |
| 19 | 13 | DC3 | 45 | 2D | - | 71 | 47 | G | 97 | 61 | a | 123 | 7B | { |
| 20 | 14 | DC4 | 46 | 2E | . | 72 | 48 | H | 98 | 62 | b | 124 | 7C | \| |
| 21 | 15 | NAK | 47 | 2F | / | 73 | 49 | I | 99 | 63 | c | 125 | 7D | } |
| 22 | 16 | SYN | 48 | 30 | 0 | 74 | 4A | J | 100 | 64 | d | 126 | 7E | ~ |
| 23 | 17 | ETB | 49 | 31 | 1 | 75 | 4B | K | 101 | 65 | e | 127 | 7F | DEL |
| 24 | 18 | CAN | 50 | 32 | 2 | 76 | 4C | L | 102 | 66 | f | | | |
| 25 | 19 | EM | 51 | 33 | 3 | 77 | 4D | M | 103 | 67 | g | | | |

# 4- Primitive Types: Booleans

- A **boolean** value represents a true or false expression.
- The reserved words ***true*** and ***false*** are the only valid values for a boolean type.
- We cannot use **0** and **1** as boolean values.