

OOP Using JAVA

04 – Basic Java syntax

Dr. Mostafa A. El-Hosseini

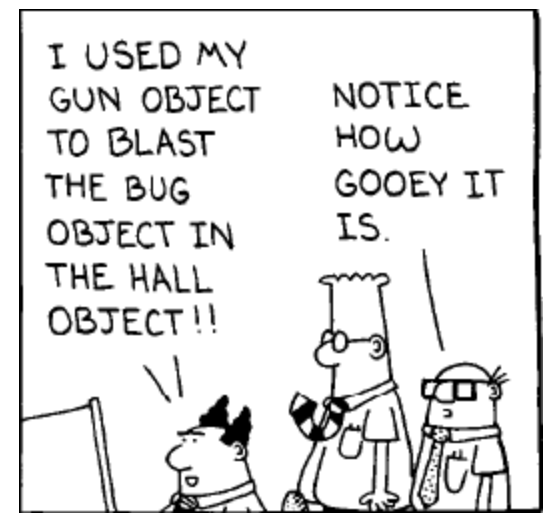
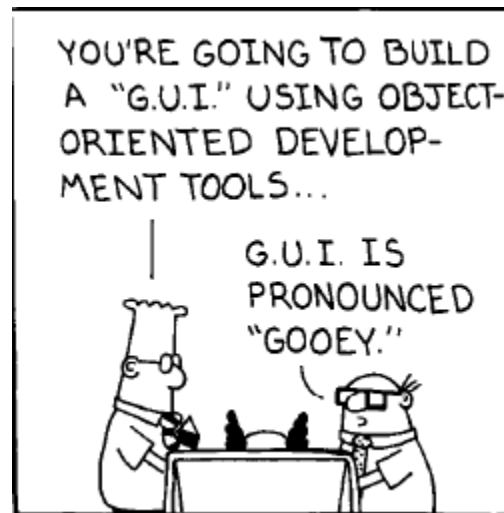
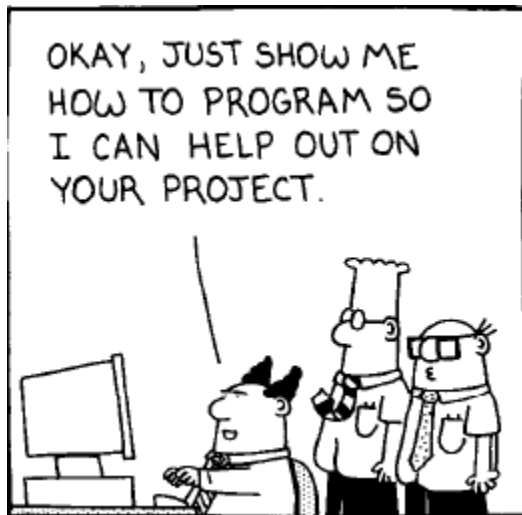
melhosseini@eng.kfs.edu.eg

RECAP

- » Library
- » Java Specification
- » API – Application Programming Interface
- » Keywords
- » Identifiers
- » Comments
- » Your First Java Program

AGENDA

Java is simple



Java is Powerful?

- » **Truth: Java has a rich set of standard libraries**
 - » Networking
 - » Threads (lightweight processes)
 - » Distributed objects
 - » Database access
 - » Graphics: GUI controls and drawing
 - » Data structure library
 - » Arbitrary precision integral and fixed-point arithmetic
 - » Digital signatures
 - » Serialization (transmitting/reassembling data structures)
 - » File and stream compression
 - » XML parsing

Creating and Running Program: Manual Approach

» Create the File

- » Write and save a file (say **Test.java**) that defines
public class Test

- » File and class names are **case sensitive**

» Compile the program

- » Compile Test.java through

- » **> javac Test.java**

- » This step creates a file called **Test.class**

» Run the program

- » For a stand-alone application, run it with

- » **> java Test**

- » Note that the command is **java**, not **javac**, and that you refer to **Test**, not **Test.class**

» To run applets

- » Load the HTML page that refers to it

Using a Smart Editor or IDE

Manual

- » Create file
 - » Notepad or any editor
- » Compile file
 - » Open Dos Windows
 - » Type `javac myclass.java`
- » Execute program
 - » Open dos windows
 - » Type `java myclass`

Editor or IDE

- » Create file
 - » Editor such as TextPad, UltraEdit, or jEdit
 - » IDE such as Eclipse IDEA, JBuilder, or NetBeans
- » Compile file
 - » Build/Make/Compile from within editor
 - » Many IDEs (e.g., Eclipse) compile automatically when file is saved
- » Execute program
 - » Run from within editor/IDE

Basic Mathematical Routines

» Very simplest routines use built-in operators

» $+$, $-$, $*$, $/$, $^$, $\%$

» Be careful with $/$ on int and long variables

» Static methods in the Math class

» So you call `Math.cos(...)`, `Math.random()`, etc.

» Most operate on double precision floating point numbers

» Simple operations: `Math.pow()`, etc.

» `pow(xy)`, `sqrt( $\sqrt{x}$ )`, `log(loge)`, `log10`

Basic Mathematical Routines

- » Trig functions: `Math.sin()`, etc.
 - » sin, cos, tan, asin, acos, atan
 - » Args are in radians, not degrees, (see `toDegrees` and `toRadians`)
- » Rounding and comparison: `Math.round()`, etc.
 - » round, floor, ceiling, abs, min, max
 - » Random numbers: `Math.random()`
- » random (`Math.random()`)
 - » returns from 0 inclusive to 1 exclusive).
 - » See Random class for more control over randomization.

Identifiers

- » Java identifiers are used to name **data items, classes, and methods**.
- » Legal characters for identifiers are the **letters and digits of the Unicode character** set, as well as the symbols **\$** and **_**.
- » The **space is not a legal character** in an identifier since Java uses the space to mark the end of names.
- » An identifier **cannot start with a digit**. The length of an identifier is virtually unlimited, although it is a good idea to keep identifiers to less than 30 characters.
- » An identifier **cannot be one of the special keywords** used by the Java language.

» The following are legal identifiers in Java:

 **personalName**

 **PI**

 **y_121**

 **\$\$128**

 **user_address**

» The following identifiers are illegal:

 **1_value**

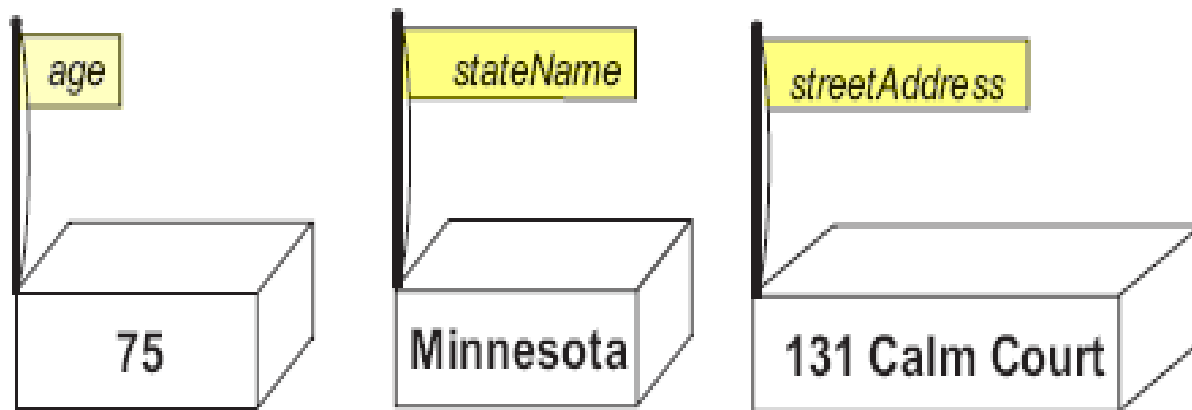
 **User name**

 **%%123**

Programmers note

- » A good identifier is **one that is descriptive** and at the same time **easy to type**.
- » For example, if a data item is to hold the age of an employee, a good name for this item may be ***employeeAge***.
- » Since spaces are illegal in identifiers, you can use **capital letters or the underscore symbol to separate the individual words in a name**.

Creating and using variables and constants



Creating Variables

- » To a programmer, a **variable is a storage location** that contains a data item of a specific type.
- » The storage location **is assigned a name** so that it can be identified.
- » The contents of a **variable can be changed** anywhere in the program.
- » **Constants**, on the other hand, represent values **that do not change in the course of a program**.
- » For example, we can define a constant to store the value of **Pi**, which is the ratio between the diameter and the circumference of a circle.
- » Once defined, this value can be recalled any time that it is needed

Java variable types

Java Primitive Data Types

	TYPE NAME	STORAGE SPACE	APPROXIMATE RANGE
INTEGRALS:	int	4 bytes	+/- 2 billion
	short	2 bytes	+/- 32,767
	long	8 bytes	+/- 9.2 x 10 ¹⁸
	byte	1 byte	-129 to 127
FLOATING-POINT:	float	4 bytes	7-8 digits
	double	8 bytes	16-17 digits
CHARACTER TYPE:	char	2 bytes	65,536
BOOLEAN TYPE:	boolean		true and false

Declaring a variable

- » Before you can use a variable in a Java program **you must first declare** it by specifying its type and assigning to it a name.
- » The **variable name must be a legal Java identifier**
- » The **purpose** of the declaration statement is to inform the compiler of our intention of creating a variable of a specific type and tagging it with a particular name.
- » The result is that **Java reserves a memory space** in which to store the data and associates it with the identifier that we assigned as a name.

Declaration Examples

- » **int age1;**
- » **int age2, age3, age4;**
- » **float radius = 1.22;**
- » **int valA = 1, val2, val3, val4 = 77;**

Java strings

- » Programs frequently use **groups of characters** called strings.
- » A **Java string is a class**, not a primitive type.
- » The Java **String** class is part of the **java.lang** library that is directly accessible to any Java program.
- » For this reason you can **create objects of the Java String class** as if they were variables of a primitive data type.
- » Note that because **String is a class it begins with an uppercase letter**, while all the other primitive data types in the Table have names that start with a lower-case letter

Java Strings

- » `String uName = "Mansoura - Dakahilia";`
- » `String ssn = "12-010-15";`
- » The Java **String** class **contains several methods** to manipulate and convert strings
- » The **+** operator has a double life outside the world of mathematics. It can **concatenate** two or more strings.
- » **Concatenate** means to link two things together
- » `String firstName = "Mohamed";`
- » `System.out.println("Everybody loves " + firstName);`

Data Input / Integer

```
» import java.util.Scanner;  
» // create Scanner to obtain input from command window  
» Scanner input = new Scanner( System.in );  
  
» int number1;      // Integer number  
» number1 = input.nextInt();    // read first number
```

Data Input / Double

```
» import java.util.Scanner;  
» // create Scanner to obtain input from command window  
» Scanner input = new Scanner( System.in );  
  
» double number1;      // Double number  
» number1 = input.nextDouble(); // read first number
```

Data Input / Char

```
» Scanner input = new Scanner( System.in );  
» String mystring = input.next();  
» char c = mystring.charAt( 0 );
```

Sample Program

```
import java.util.Scanner;
class mycircle
{
    public static void main(String args[])
    {
        double pi = 3.14;
        Scanner input = new Scanner(System.in);
        System.out.print("Raidus= ");
        double r = input.nextDouble();
        double area = pi*r*r;
        System.out.printf("Area = %f",area);
    }
}
```