Fundamentals of Programming 1

Lecture 2: Variables + Arithmetic Operators

Re-cap Lecture 1

- Programming, the bigger picture: involves ***problem solving*** and ***algorithms***
  - *Algorithm: is a finite, step-by-step procedure for accomplishing some task or solving a problem*
- A ***computer program*** or ***application*** is a set of instructions (algorithm!), written in a programming language that enables a computer to perform some specified task.
- A computer executes these instructions encoded in its own unique native ***machine language*** (machine code). To do this...
  - The programming language (***source code***) must be translated into the machine language of that computer
  - Translation is the job of a program called a ***compiler***
  - Once a compiler translates the source program into machine language the computer can execute the resulting target program
- Compilation – traditional way (C++,C...): one compiler for Windows, another to translate it for a Mac i.e. need a different compiler to translate the program into the machine language for a computer running different OS but just one program

Re-cap Lecture 1

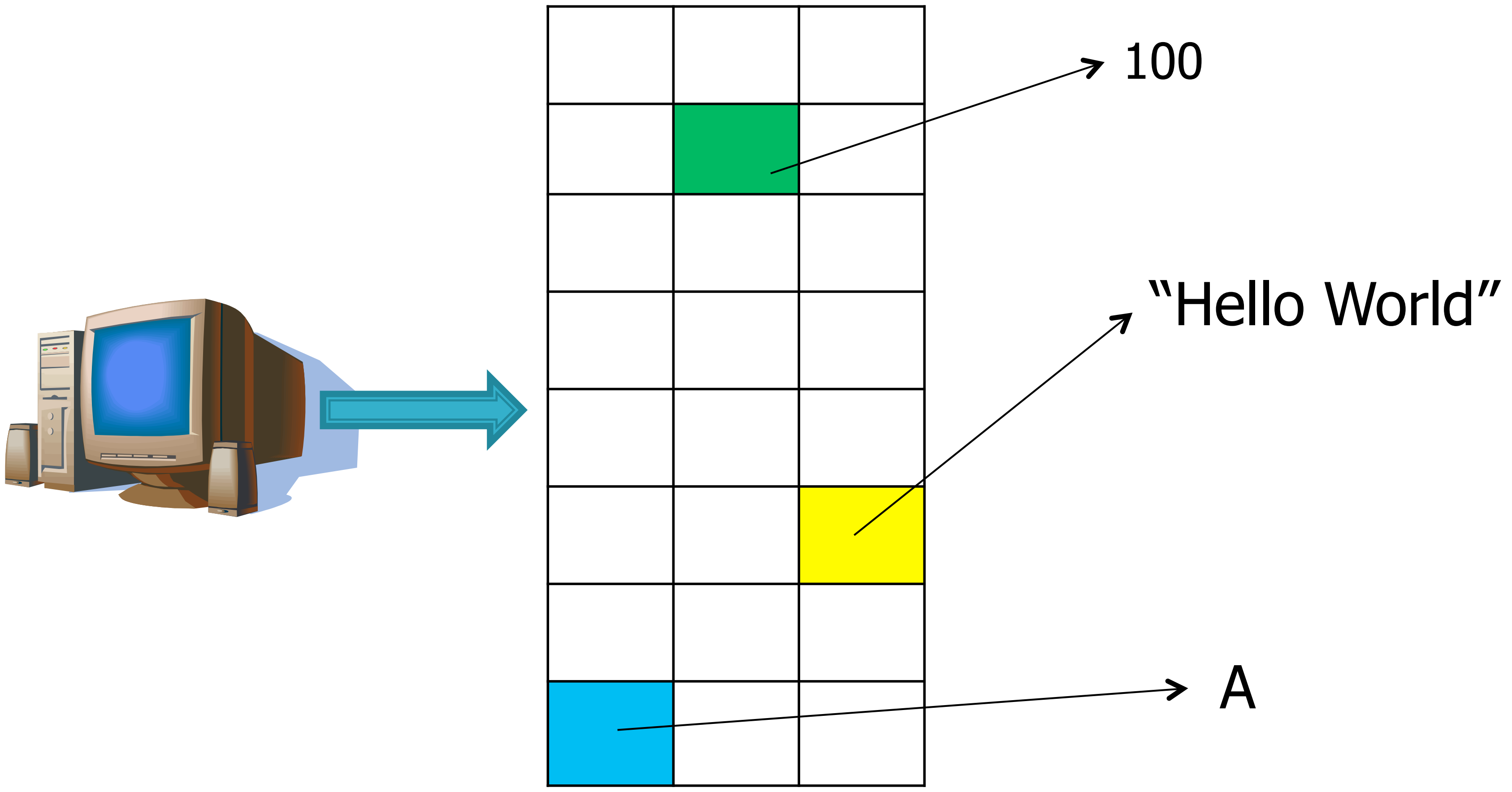
- Then, along came **Java** (the language we will use to write our programs) – Java is ***platform independent.*** This adds a second stage to the compilation process (compilation + interpretation) and means that your Java program can run/be executed on any OS without needing a different compiler for each machine type. To make Java a cross-platform programming language use ***JVM*** **(*Java Virtual Machine*).** The Java program is compiled into java ***bytecode***, and can be run on any machine with the JVM installed. So we have an extra step/layer between source code and execution.
- How to compile Java Programs – different **IDEs (Integrated Development Environments)**, each with a slick graphical interface that provide an editor for writing programs, file browsing, debugger, compilation and execution e.g. Eclipse, Jcreator...
- You do not need an IDE to write and run programs – can write your program using a simple text editor – **TextPad** – and compile your program using the **Java Development Kit (JDK)** which you can download free from ORACLE
- We wrote our first program – **class** definition + program name, **main method** declaration, and **print** and **println** to display text to the screen

Lecture 2 Objectives

- Declaring Variables
- Understand the different data types used in programming
- Understand how to choose suitable identifiers for your variables
- Assign values to variables
- Arithmetic Operators and Precedence
- Performing calculations with variables

Variables in Java

How memory works



Variables

- If we want our programs to store information while they run, we need to set aside space in memory
- We tell it what type of data (or information) we wish to store…computers can’t handle infinity…need limits and ranges!
- We essentially set aside a little block of memory for each piece of data we wish to store
- We can change or ‘vary’ the values of this memory, thus the name ‘variable’

Java data types

The data types of Java		
Java type	**Allows for**	**Range of values**
`byte`	very small integers	-128 to 127
`short`	small integers	-32768 to 32767
`int`	big integers	-2147483648 to 2147483647
`long`	very big integers	-9223372036854775808 to 9223372036854775807
`float`	real numbers	+/- 1.4 * 10^{-45} to 3.4 * 10^{38}
`double`	very big real numbers	+/- 4.9 * 10^{-324} to 1.8 * 10^{308}
`char`	characters	Unicode character set
`boolean`	true or false	not applicable

http://download.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Declaring a variable

- the above data types are used in programs to specify the type of a variable and thus the values it can store

- this process is known as **declaring**

- these named locations are called **variables.**

- we can choose any name for variables as long as:
  - the name is not already a word in the Java language i.e. a reserved word (such as **class, void**);
  - the name has no spaces in it;
  - the name does not include operators or mathematical symbols such as + and –;
  - the name starts either with a letter, an underscore (_), or a dollar sign ($).

- the convention in Java programs is to begin the name of a variable with a *lower case* letter.

Reserved Words

High Level Languages, including Java, are usually defined in terms of a set of **reserved** words. The Java programming language is defined in terms of around 40 reserved words, which make up most of the basic command vocabulary of the language. See the full list at the link below:

http://download.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Reserved words in Java

exit
break
void
if
main
case
end
system
double
do
else
(etc…)

Note:
These reserved words cannot be redefined for other uses: i.e. they may not be used as for variables names.

Declaring a Variable

- To declare a variable we must
  - Give that variable a name (you decide... adhering to rules outline in previous slides!)
  - Decide on the data type that the variable will store

→ int, double, char, boolean

Declaring a variable: an example

Let’s create a variable to keep a player’s score in a computer game.

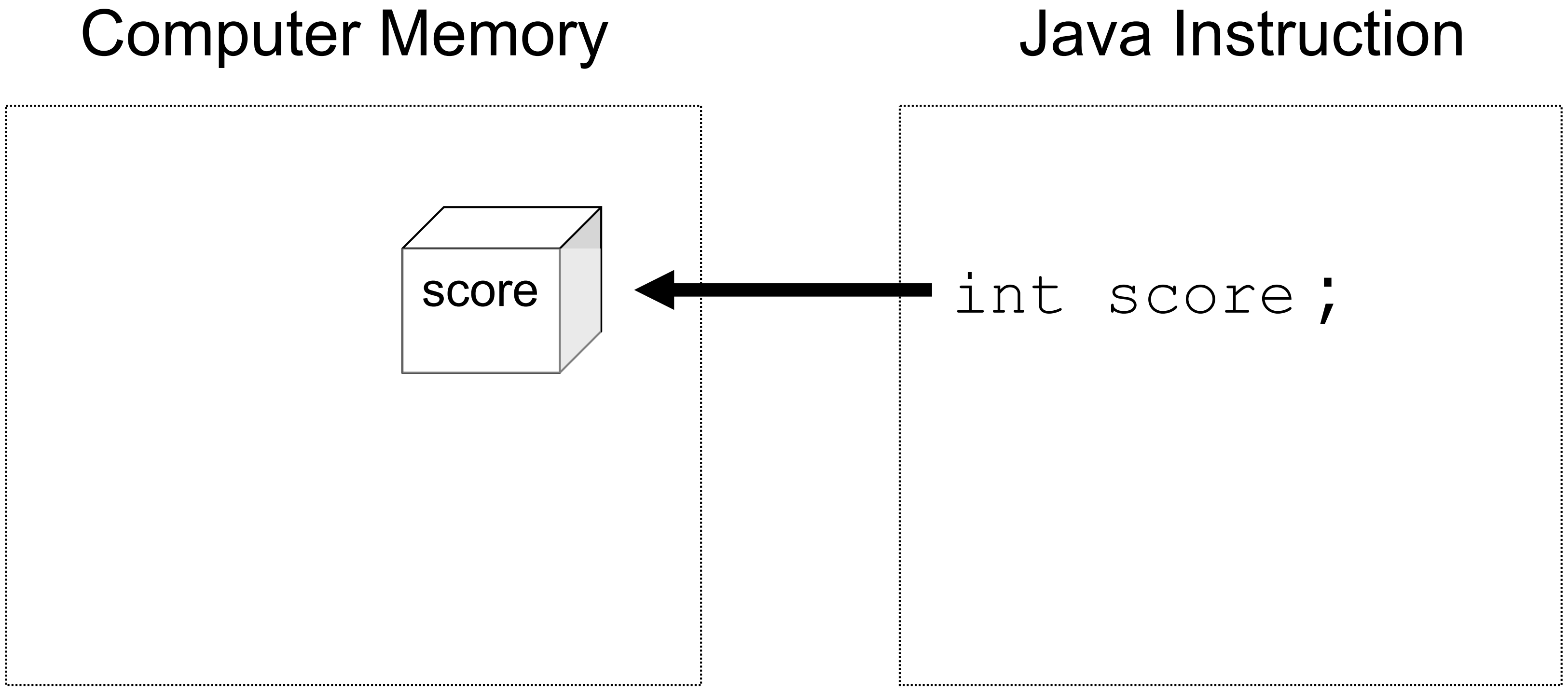
```
dataType variableName;
```

a score will always be a whole number

good meaningful name

```
int score ;
```

The effect of declaring a variable in Java



```
int score;
```

Declaring many variables

Assume that the player of a game can choose a difficulty level (A, B, or C).

```
int score; // to hold score

char level; // to hold difficulty level
```

Declaring variables of the same type

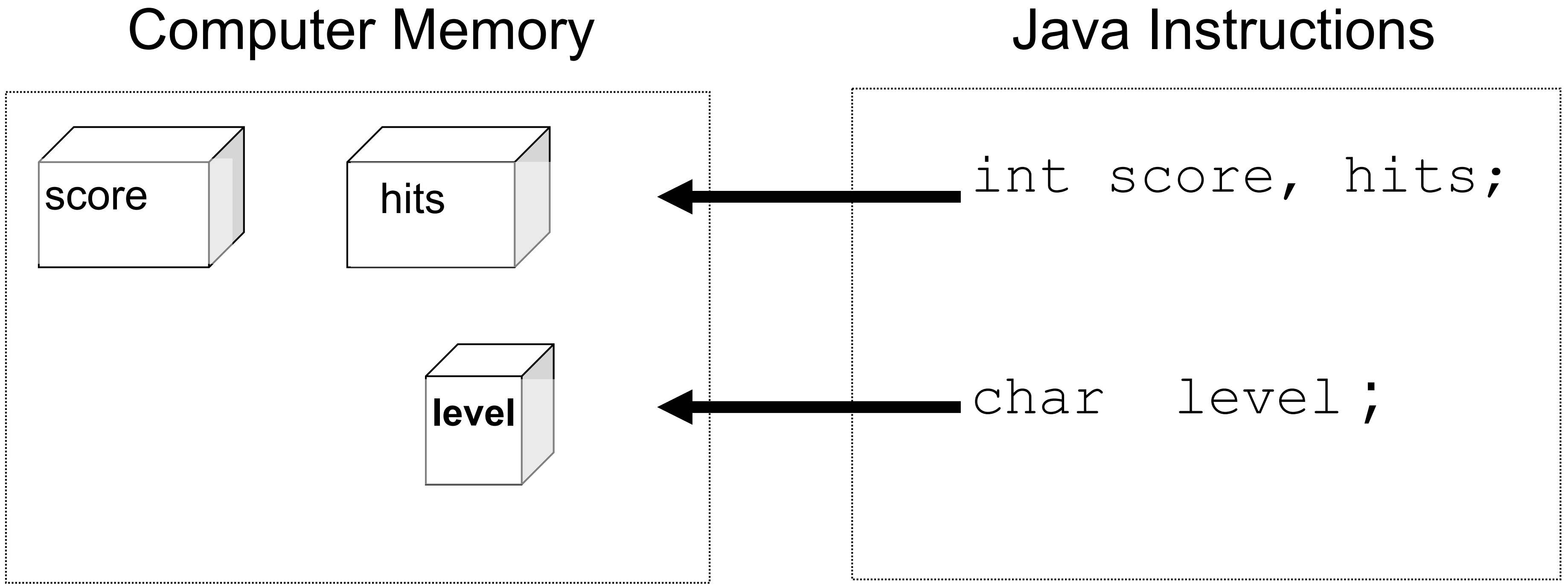
Several variables can be declared on a *single line* if they are *all of the same type*.

Assume that there are ghosts in the house that hit out at the player; the number of times a player gets hit by a ghost can also be recorded.

```
int score, hits; // both the same type

char level ;  // different type
```

The effect of declaring many variables in Java



Activity: data types

- A mark between 0–100 (whole number, no decimal)
- How could we declare this variable in our program?

- Students grade – A, B, C, D, E, or F.
- How could we declare this variable in our program?

- Weight of a food item purchased in a supermarket.
- How could we declare this variable in our program?

Assignments in Java

Assignments allow values to be put into variables.

Written in Java with the use of the equality symbol (=), known as **the assignment operator**.

Simple assignments take the following form:

```
variableName = value;
```

```
score = 0;
```

Initializing variables

You may combine the assignment statement with a variable declaration to put an initial value into a variable as follows:

```
      int score = 0;
```

Note, the following declaration will **not** compile in Java:

```
      int score = 2.5 ;
```

This will not compile because 2.5 is a **double** value

Putting values into character variables

When assigning a value to a character variable, you must enclose the value in single quotes.

for example

set the initial difficulty level to A

```
      char level = ‘A’;
```

Re-assigning variables

Remember: you need to declare a variable only once.

You can then assign to it as many times as you like.

```
char level = ‘A’;
level = ‘B’;
```

Creating constants

Constants are data items whose values *do not change*. For example:

- the maximum score in an exam (100);
- the number of hours in a day (24);
- the mathematical value of π (3.1417).

Constants are declared much like variables except

```
    final int HOURS = 24;
```

- they are preceded by the keyword **final**
- they are always initialised to their fixed value.

Activity: data types & assignment

- Explain which, if any, of the following would result in a compiler error:
  - int x = 75.5;
  - double y = 75;
  - char grade = A;

Another example of assignment in Java

Assigning a Variable

Once the declaration is made with a legal name (identifier), operations can be performed on the variable, e.g.

```
int age;
age = 21;
System.out.print(age);
```

This command displays the Integer age to the screen

Assignment of variables

An assignment of a value to a variable can be made in the following format:

```
x = 100;
```

An assignment to an integer variable can also be made by reading from the user. This is achieved in the following manner

```
x = Keyboard.readInt( );
```

where x is the variables name

Example

- A proper program segment, including a character output to prompt the user, would look like,

```
{
   int age;
   System.out.println("Enter Age : ");
   age = Keyboard.readInt( );
 }
```

*****Note: The Keyboard class takes care of input; you will need to put Keyboard.class into the directory where you are working*

Complete Program

```
class getage
{
    public static void main(String[ ] args)
    {
        int age;
        System.out.println(“Enter your Age:     “);
        age = Keyboard.readInt( );
        System.out.println(“Your age is  “);
        System.out.println(age);
    }
}
```

- Run program…

Activity: reading from the keyboard

- Consider the following declaration of a variable to store students examination marks:

  double mark;

  - Which one of the following statements allows the user to enter a mark of 70 into this variable while the program is running?
    a) mark = 70;
    b) mark = System.out.print(“Please enter a mark”);
    c) mark = Keyboard.readDouble();
    d) mark = Keyboard.readInt();
    e) mark = “70”;

Re-assigning variables

Once the user has read a variable in, it can still be assigned another value, e.g.

```
 int   age;
age = Keyboard.readInt( );
age = 17;             //Assignment Statement
System.out.println("Your age is:   ");
System.out.println(age);
```

Run program

It is overwritten.

- No matter what you enter as your age using the Keyboard.readInt( ) command the output here will always be "Your Age is: 17"

Swapping Algorithm

- A Program to swap the employee number of two employees. Is it successful?

```
{
  int  tom   = 1200;
  int  alice  = 2100;

 Alice = Tom;
 Tom = Alice;
 Screen.writeint(tom);
 Screen.writeint(alice);
}
```

Run program...

Use a temporary variable.

- No it is not, both will end up with the same value.

  The introduction of a temporary variable to store the original contents of tom would be the most logical solution:

```
{
  int  tom    = 1200;
  int  alice  = 2100;
  int temp;

  temp = tom;
  tom = alice;
  alice = temp;

  System.out.println(tom);
  System.out.println(alice);
}
```

This is a common procedure. You’ll see it again in the future.

Run program...

Calculations on Variables

Sample:Program to calculate the sum of two numbers.

```
class sum{
   public static void main(String[ ] args){

      int first, second, result;

      System.out.print("Enter first number");
      first = Keyboard.readInt( );

      System.out.print("Enter second number");
      second= Keyboard.readInt( );

      result = first  +  second;
      System.out.print("The Result is  :    ");
      System.out.print(result);
   }
}
```

Terms on RHS referred to an **expressions**

You can use the name of the variable you are assigning to in the expression itself

- This means that the old value of the variable is being used to calculate the new value

Example: first = first + second

The new value of 'first'

The old value of 'first'

Arithmetic Operators

Operations that can be carried out on data include:

+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Modulus (remainder Function)

Operator Precedence

The order of precedence for arithmetic operators is:

( )	*Brackets*
***	*Multiplication*
/	*Division*
+	*Addition*
-	*Subtraction*

Comments and Documentation

Comments may be added to a Java program to describe what each line or command is doing. This is done by placing two slashes // before any comments you wish to make as documentation within the program.

e.g.
```
result = first + second;               //  Evaluating Sum
System.out.println("Result is ");      //  Output Sum
System.out.println(result);            //  To the screen
```

Documentation

<u>**Documentation**</u> is a set of text notes that appear within a program. These notes include a description of how the program is used (User's Guide).

The purpose of the program, how it is installed, tested and expected results. Other important information to be included is the name of the person(s) who wrote the program and the date on which it was written.

Importance of Documentation

NOTE: The importance of adequate documentation cannot be over emphasised. if a program is to receive regular use it is likely that some modification will eventually be necessary.

This is difficult even for the original author unless there is sufficient information available about the program.

DON'T OVER DOCUMENT ! !

Guide to Requirements

eg:

```
/* **********************************************************/
/*   Java Program to add two numbers.                      */
/*   Written by:         .......................................      */
/*  Course            :.......................................      */
/*  Language          :.......................................      */
/*  Date Started      :.......................................      */
/*  Last Update       :.......................................      */
/*  Program Description       :...........................      */
/*       ..........................................................    */
/*       ..............................................................*/
/**********************************************************/
Class MyProg{
                 Etc….
```

Indentation

Programs should be indented consistently;

ie: the number of spacings to the right for each command must be appropriate and the same for each similar section of code.

Activity: calculations on variables

- Program 1:
  - A class of 100 students have been told to set themselves up into 6 groups for their course work. Write a program that calculates and displays how many students will be in each group and how many students without a group. Declare all variables necessary.
- Program 2:
  - A class of students have been told to set themselves up into groups for their course work. Write a program that prompts the user for the number of students in the class and the number of groups to be formed and calculates and displays how many students will be in each group and how many students without a group. Declare all variables necessary.