

Computer Programming I - CSC111

Chapter 2 – Basic Computation

Dr. Mejd I Safran

mejdl@ksu.edu.sa

Outline

- Java program structure
- Hello program
- Saving, compiling and running Java programs
- Comments
- What is a variable?
- What is a data type?
- Basic data types
- Variables identifiers
- How to select proper types for numerical data
- Assignment statement
- Simple keyboard input/output

Java Program Structure

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

Java Program Structure

```
// import section - import used Java libraries
public class MyProgramName {

    // main method
    public static void main(String[] args) {

        // declaration section - declare needed variables

        // input section - enter required data

        // processing section - processing statements

        // output section - display expected results

    } // end main
} // end class
```

Hello Program

```
// import section - import used Java libraries
public class HelloProgram {

    // main method
    public static void main(String[] args) {

        // declaration section - declare needed variables

        // input section - enter required data

        // processing section - processing statements

        // output section - display expected results

        System.out.println("... Hello ...");

    } // end main

} // end class
```

Saving, compiling, and running Java program

- Saving a Java program

- A file having a name same as the class name should be used to save the program. The extension of this file is “.java”

- Compiling a Java program

- Call the Java compiler **javac**:
 - javac HelloProgram.java
 - generates a file called “HelloProgram.class” (the bytecode)

- Running a Java program

- Call JVM **java**:
 - java HelloProgram

Comments

- The best programs are self-documenting
 - Clean style
 - Well-chosen names
- Comments are written into a program as needed to explain the program
- They are useful for programmer, but they are ignored by the compiler

Comments

- A comment can begin with //
- Everything after these symbols and to the end of the line is treated as a comment

```
double radius; // in centimeters
```


Comments

- A comment can begin with `/*` and end with `*/`
- Everything between these symbols is treated as a comment

```
/*  
  This is an example of  
  multiple lines of comments  
*/
```

Hello Program

```
// import section - import used Java libraries
public class HelloProgram {

    // main method
    public static void main(String[] args) {

        /* declaration section
           In this section you should declare needed variables
        */

        // input section - enter required data

        // processing section - processing statements

        // output section - display expected results

        System.out.println("... Hello ...");

    } // end main
} // end class
```

Comments

- A *javadoc* comment, begins with `/**` and end with `*/`
- It can be extracted automatically from Java software

```
/** method change requires the number of coins to be  
    nonnegative */
```

When to use comments

- Begin each program file with an explanatory comment
 - What the program does
 - The name of the author
 - Contact information for the author
 - Date of the last modification
- Provide only those comments which the expected reader of the program file will need in order to understand it.

Sum program

```
public class SumProgram {  
    public static void main(String[] args) {  
        int num1;  
        int num2;  
        int sum;  
  
        num1 = 4;  
        num2 = 3;  
  
        sum = num1 + num2 ;  
  
        System.out.println(sum);  
    }  
}
```

Variables

- *Variables* are used to store (represent) data such as numbers and letters.
 - Think of them as places to store data.
 - They are implemented as memory locations.
- The data stored by a variable is called its *value*.
 - The value is stored in the memory location.
- Its value can be changed.

Variables and values

- Variables:

`numberOfStudents`

`studentGPA`

`age`

- Assigning values:

`numberOfStudents = 100 ;`

`numberOfStudents = numberOfStudents - 1 ;`

Naming and declaring variables

- Choose names that are helpful such as **count** or **speed**, but not **c** or **s**.
- When you *declare* a variable, you provide its name and type

- without initial value

```
int count, sum;
```

```
double avg;
```

- with initial value

```
int count = 5, sum = 200;
```

```
double avg = sum / count;
```

- A variable's *type* determines what kinds of values it can hold (**int**, **double**, **char**, etc.).
- A variable must be declared before it is used.

Syntax and examples

- Syntax

```
type variable_1, variable_2, ...;
```

- Examples

```
int styleChoice, numberOfChecks;  
double balance, interestRate;  
char jointOrIndividual;
```

Data types in Java

- In Java, you have to specify a *data type* of variables that tells what kind of data to be stored in it.
- Two kinds of data types:
 - 1) *Primitive data types*: for single number or single letter, e.g. **int**
 - 2) *Class types*: for objects of a class, e.g. class type **String**

Primitive types

- Four integer types (**byte**, **short**, **int**, and **long**)
 - **int** is most common
- Two floating-point types (**float** and **double**)
 - **double** is more common
- One character type (**char**)
- One Boolean type (**Boolean**)

Primitive data types

Type Name	Kind of Value	Memory Used	Range of Values
byte	Integer	1 byte	−128 to 127
short	Integer	2 bytes	−32,768 to 32,767
int	Integer	4 bytes	−2,147,483,648 to 2,147,483,647
long	Integer	8 bytes	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$
double	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
char	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
boolean		1 bit	True or false

Primitive data types

*Primitive data types never allow us to store multiple values of same type!
(one value at a time)*

```
int a;           // valid  
a = 10;          // valid  
a = 10, 20, 30;  // invalid
```

Examples of primitive values

- Integer types

0 -1 365 12000

- Floating-point types

0.99 -22.8 3.14159 5.0

- Character type

'a' 'A' '#' ' ' '

- Boolean type

true false

Java identifiers

- The name of something in a java program such as a variable, class, or method, is called an *identifier*.
- Identifiers may contain only
 - Letters
 - Digits (0 through 9)
 - The underscore character (_)
 - And the dollar sign symbol (\$) which has a special meaning (*avoid it*)
- The first character cannot be a digit.

Java identifiers

- Identifiers may not contain any spaces, dots (.), asterisks (*), or other characters:

7-11 **netscape.com** **util.*** (not allowed)

- Identifiers can be arbitrarily long.
- Since Java is *case sensitive*, **stuff**, **Stuff**, and **STUFF** are different identifiers.

Keyword or reserved words

- Words such as **if** are called *keywords* or *reserved words* and have special, predefined meanings.
 - Cannot be used as identifiers.

Keywords in Java				
abstract	default	if	private	this
assert	do	implements	protected	throw
boolean	double	import	public	throws
break	else	instanceof	return	transient
byte	enum	int	short	try
case	extends	interface	static	void
catch	final	long	strictfp	volatile
char	finally	native	super	while
class	float	new	switch	
continue	for	package	synchronized	

Constant declaration

- A **constant** is a variable whose value cannot change once it has been assigned.
- Syntax

```
final type constIdentifier = literal | expression;
```
- Examples

```
final double PI = 3.14159;
final int MONTH_IN_YEAR = 12;
final int MAX = 1024;
final int MIN = 128;
final double AVG = (MAX + MIN) / 2;
```

Naming conventions

- Class types begin with an uppercase letter (e.g. `String`).
- Primitive types begin with a lowercase letter (e.g. `int`).
- Variables:
 - All lowercase letters, capitalizing the first letter of each word in a multiword identifier, except for the first word (e.g. `myName`, `myBalance`).
- Constants:
 - All uppercase letters, separating words within multiword identifier with the underscore symbol (e.g., `IP`, `MONTH_IN_YEAR`).

Where to declare variables

- Declare a variable
 - Just before it is used or
 - At the beginning of the section of your program that is enclosed in `{ }`.

```
public static void main(String[] args)
{ /* declare variables here */
    . . .
}
```

Select proper types for numerical data

- Define the following variables for a phone application:
 - Number of users (**int**)
 - Number of photos (**int**)
 - Number of comments (**int**)
 - Average number of active users in a week (**double**)
 - Is the app available? (**boolean**)
 - Version (**float or double**)
 - User's gender (**char**)
 - Account balance (**double**)
 - Max number of photos allowed (**final int**)

Assignment statements

- An assignment statement is used to assign a value to a variable.

```
answer = 42;
```

- The "equal sign" is called the *assignment operator*.
- We say, "The variable named **answer** is assigned a value of 42," or more simply, "**answer** is assigned 42."

Assignment statements

- Syntax

variable = expression

where **expression** can be another variable, a *literal* or *constant* (such as a number), or something more complicated which combines variables and literals using *operators* (such as + and -)

Assignment examples

```
amount = 3.99;  
firstInitial = 'W';  
score = numberOfCards + handicap;  
eggsPerBasket = eggsPerBasket - 2;
```


Assignment evaluation

- The expression on the right-hand side of the assignment operator (=) is evaluated first.
- The result is used to set the value of the variable on the left-hand side of the assignment operator.

```
avg = sum / count;  
counter = counter - 2;
```

Initializing variables

- A variable that has been declared, but no yet given a value is said to be *uninitialized*.
- Uninitialized class variables have the value **null**.
- Uninitialized primitive variables may have a default value.
- It's good practice not to rely on a default value.

Initializing variables

- To protect against an uninitialized variable (and to keep the compiler happy), assign a value at the time the variable is declared.
- Examples:

```
int count = 0;
```

```
char grade = 'A';
```

Initializing variables

- syntax

```
type variable_1 = expression_1,  
variable_2 = expression_2, ...;
```

Simple input

- Sometimes the data needed for a computation are obtained from the user at run time.

- Keyboard input requires:

`import java.util.Scanner`

at the beginning of the file

```
public class SumProgram {  
  
    public static void main(String[] args) {  
  
        int num1;  
        int num2;  
        int sum;  
  
        num1 = 4;  
        num2 = 3;  
  
        sum = num1 + num2 ;  
  
        System.out.println(sum);  
    }  
}
```

Simple input

- Data can be entered from the keyboard using

```
Scanner keyboard = new Scanner(System.in);
```

followed by

```
num1 = keyboard.nextInt();
```

which read one `int` value from the keyboard and assign it to `num1`

Simple input

```
import java.util.Scanner;

public class SumProgram {

    public static void main(String[] args) {

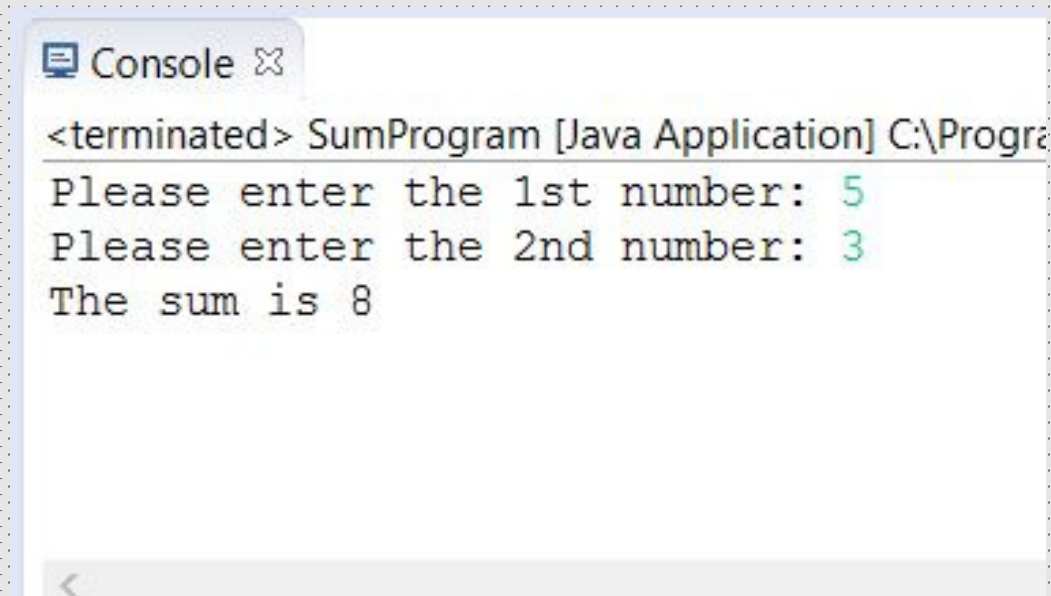
        Scanner keyboard = new Scanner(System.in);
        int num1;
        int num2;
        int sum;

        System.out.print("Please enter the 1st number: ");
        num1 = keyboard.nextInt();
        System.out.print("Please enter the 2nd number: ");
        num2 = keyboard.nextInt();

        sum = num1 + num2 ;

        System.out.println("The sum is " + sum);
    }
}
```

Simple screen output



```
Console X
<terminated> SumProgram [Java Application] C:\Progra
Please enter the 1st number: 5
Please enter the 2nd number: 3
The sum is 8
```

The screenshot shows a console window titled "Console" with a close button. The window displays the output of a Java application named "SumProgram". The output consists of three lines: "Please enter the 1st number: 5", "Please enter the 2nd number: 3", and "The sum is 8". The numbers 5 and 3 are highlighted in green, indicating user input. The window title bar shows the application path as "C:\Progra".

Common Scanner methods

Method	Example
<code>nextByte()</code>	<code>byte b = input.nextByte();</code>
<code>nextShort()</code>	<code>short s = input.nextShort();</code>
<code>nextInt()</code>	<code>int i = input.nextInt();</code>
<code>nextLong()</code>	<code>long l = input.nextLong();</code>
<code>nextFloat()</code>	<code>float f = input.nextFloat();</code>
<code>nextDouble()</code>	<code>double d = input.nextDouble();</code>
<code>next()</code>	<code>String str = input.next();</code>
<code>nextLine()</code>	<code>String str = input.nextLine();</code>

- Last two methods will be discussed and used when we study the class `String`