**Administrative**

**Today’s session**

Java online resources

C++ and Java side-by-side

Java templates

main method

Comment

println method

print method

Escape sequences

**Session Topics**

**Java online resources**

● A specification of the Java language is available in document **Java language specification** on Blackboard.

● The Java API specification is available at [docs.oracle.com/javase/8/docs/api/](http://docs.oracle.com/javase/8/docs/api/).

● Java tutorials are available at:

[docs.oracle.com/javase/tutorial/index.html](http://docs.oracle.com/javase/tutorial/index.html)

[www.tutorialspoint.com/java/](https://www.tutorialspoint.com/java/)

[www.javatpoint.com/java-tutorial](https://www.javatpoint.com/java-tutorial)

**C++ and Java side-by-side**

● The basic elements of the C++ and Java languages are very similar.

● A comparison of the C++ and Java languages is available in document **C++ and Java Side-by-side** on Blackboard.

● Here is a comparison of a factorial calculator written in the C++ and Java languages:

**C++ application version**

//==========================================================

//

// Title: Factorial Calculator

// Description:

// This application prompts for and reads a whole number

// from the user until it is between 0 and 170. It then

// calculates and display its factorial.

//

//==========================================================

#include <conio.h> // For function getch()

#include <iostream> // For cin, cout, and system

using namespace std; // So "std::cout" may be abbreviated to "cout", for example.

int main()

{

// Declare variables

int number;

double fact; // Declared 'double' to hold larger numbers

// than 'int'

// Show application header

cout << "Welcome to Factorial Calculator" << endl;

cout << "-------------------------------" << endl << endl;

// Loop to get number.

// -Factorial is not defined for numbers less than 0.

// -double cannot hold number larger than 170!.

cout << "Enter a whole number to calculate the factorial "

<< "of (between 0 and 170): ";

cin >> number;

while (number < 0 || number > 170)

{

cout << "\nEnter a whole number to calculate the "

<< "factorial of (between 0 and 170): ";

cin >> number;

}

// Test which number entered

if (number == 0 || number == 1)

fact = 1;

// Calculate factorial of numbers greater than 1

else

{

// Loop to calculate factorial

fact = 1;

for (int i = number; i > 1; i--)

fact = fact \* i;

}

// Show factorial

cout << number << "! = " << fact << endl;

// Show application close

cout << "\nEnd of Factorial Calculator" << endl << endl;

// Pause before application window closes

cout << "Press any key to exit ..." << endl;

\_getch();

}

**Java application version**

**//==============================================================**

**//**

**// Title: Factorial Calculator**

**// Description:**

**// This application prompts for and reads a whole number**

**// from the user until it is between 0 and 170. It then**

**// calculates and display its factorial.**

**//**

**//==============================================================**

**package wsu.FactorialCalculator;**

**//Import classes**

**import java.util.Scanner;**

**//--------------------------------------------------------------**

**// class FactorialCalculator**

**//--------------------------------------------------------------**

**public class FactorialCalculator**

**{**

**//----------------------------------------------------------------**

**// main**

**//----------------------------------------------------------------**

**public static void main(String[] args)**

**{**

**// Declare variables**

**Scanner keyboard = new Scanner(System.in); // Connect to keyboard**

**int number;**

**double fact; // Declared 'double' to hold larger numbers**

**// than 'int'**

**// Show application header**

**System.out.println("Welcome to Factorial Calculator");**

**System.out.println("-------------------------------\n");**

**// Loop to get number.**

**// -Factorial is not defined for numbers less than 0.**

**// -double cannot hold number larger than 170!.**

**System.out.print("Enter a whole number to calculate the factorial " +**

**"of (between 0 and 170): ");**

**number = keyboard.nextInt();**

**while (number < 0 || number > 170)**

**{**

**System.out.print("\nEnter a whole number to calculate the " +**

**"factorial of (between 0 and 170): ");**

**number = keyboard.nextInt();**

**}**

**// Test which number entered**

**if (number == 0 || number == 1)**

**fact = 1;**

**// Calculate factorial of numbers greater than 1**

**else**

**{**

**// Loop to calculate factorial**

**fact = 1;**

**for (int i = number; i > 1; i--)**

**fact = fact \* i;**

**}**

**System.out.println(number + "! = " + fact);**

**// Close keyboard**

**keyboard.close();**

**// Show application close**

**System.out.println("\nEnd of Factorial Calculator\n");**

**}**

**}**

**C++ application output**

Welcome to Factorial Calculator

-------------------------------

Enter a whole number to calculate the factorial of (between 0 and 170): 50

50! = 3.04141e+064

End of Factorial Calculator

Press any key to continue . . .

**Java application output**

Welcome to Factorial Calculator

-------------------------------

Enter a whole number to calculate the factorial of (between 0 and 170): 50

50! = 3.0414093201713376E64

End of Factorial Calculator

● See **Factorial calculator (C++)** sample application on Blackboard.

● See **Factorial calculator (Java)** sample application on Blackboard.

**Java templates**

● A recommended style for Java programs is available on Blackboard:

✓ **Java template – Main class** is for your main class.

✓ **Java template – Other class** is for all other classes.

● See **Java template – Main class** on Blackboard.

● See **Java template – Other class** on Blackboard.

**main method**

● A Java application begins by executing the **main** method.

● The main method has syntax:

public static void main (String[] args)

**Where …**

**public** is an access modifier.

**static** means the method is attached to the class (not an object).

**void** means the method doesn’t return a value.

**String[] args** is a string array that contains any values passed into the program from the operating system.

● An application may contain more than one class with a main method. This is often done to test the application. After the application is tested:

✓ The extra main methods may be removed.

OR

✓ An **Eclipse run configuration** may be specified that indicates which class contains the main method that the application should begin with.

● To manage Eclipse run configurations, select **Run / Run Configurations … / Java Application**

● To manage values passed into the args array, select **Run / Run Configurations … / Java Application / <project-name> / Arguments / Program arguments**.

● To specify Eclipse variables as values passed into the args array, select **Run / Run Configurations … / Java Application / <project-name> / Arguments / Program arguments / Variables …**

● See **args array** sample application on Blackboard.

**Comment**

● The **comment** is the most important statement in any programming language.

● The comment has two forms in Java:

✓ “//” signifies a single-line comment.

✓ “/\*” starts a multi-line comment, and “\*/” ends it.

**println method**

● The **println** method sends output to the screen and then moves the output cursor to the next line.

● The println method has syntax:

**System.out.println(<expression>);**

Where …

**System** is a utility class.

**out** is a member of the System class that directs output to the “standard console”, the screen.

**println** is a member of **out**.

**<expression>** is some combination of:

String literals

Numeric literals

Method calls

Arithmetic calculations

● Two or more expression parts are connected with the **concatenation operator** (+) to create one overall expression.

● Arithmetic calculations may include the **addition operator** (+). Java distinguishes between the concatenation operator and the addition operator by reading from left to right in the statement and when a **+** is read, it tests if both sides of the **+** are numeric. If so, the **+** is treated as an addition operator.

**print method**

● The **print** method sends output to the screen but *does not* move the output cursor to the next line.

● The print method is useful in user prompts.

● The print method has syntax:

**System.out.print(<expression>);**

**Escape sequences**

● An escape sequence is a pair of characters used in a string to perform one of two operations:

✓ Enable a special character to be outputted.

✓ Move the output cursor.

● The first character is the backslash (\).

|  |  |
| --- | --- |
| Sequence | Purpose |
| \\ | Output a backslash. |
| \’ | Output a single quote. |
| \” | Output a double quote. |
| \b | Backspace cursor. |
| \f | Move cursor to start of new page. |
| \n | Start on a new line. |
| \r | Move cursor to start of current line. |
| \t | Tab on the current line. |

● See **Tab stops** sample application on Blackboard.