جامعة بيرزيت

BIRZEIT UNIVERSITY

COMP231

Advanced Programming

Welcome to COMP231,
one of the most
exciting
programming courses
offered at Computer
Science Department

Course Description

In this course, you will learn some of the concepts, fundamental syntax, and thought processes behind true **O**bject-**O**riented **P**rogramming (**OOP**)

Course Description

- Upon completion of this course, you'll be able to:
  - Demonstrate understanding of classes, constructors, objects, and instantiation.
  - Access variables and modifier keywords.
  - Develop methods using parameters and return values.
  - Build control structures in an object-oriented environment.
  - Convert data types using API methods and objects.
  - Design object-oriented programs using scope, inheritance, and other design techniques.
  - Create an object-oriented application using Java packages, APIs. and interfaces, in conjunction with classes and objects.

Logistics

- Instructor: **Ahmad Abusnaina**(Masri417)
- Text book:
  - Introduction To JAVA Programming, **10th** edition.
  - Author: Y. Daniel Liang.
  - Publisher: Prentice Hall.
- Lab Manual:
  - **Title:** LABORATORY WORK BOOK (**COMP231 Updated**)
- Eclipse

Grading Criteria

❖ Midterm exam	30%
❖ 4 Assignments	10%
❖ 4 Quizzes	15%
❖ Final Practical Exam	10%
❖ Final exam	35%

Special Regulations

- **Assignments**:
  - All assignments are **individual** efforts any duplicated copies will be treated as a cheating attempt which lead to **ZERO** mark.
  - Using code from the internet will be treated as cheating as well.
  - The assignments should be **submitted through Ritaj** within the specified deadline.
  - No late submissions are accepted even by **1 minute** after the deadline.

Special Class Regulations

- **Attendance** is mandatory. University regulations will be **strictly** enforced.
- **Mobile**: Keep it off during the class. If your mobile ring you have to leave the classroom quickly, quietly and don’t come back.
- **Late**: you are expected to be in the classroom before the teacher arrival. After **5** minutes you will not allowed entering the classroom.

Course Outline

Topics	Chapter	# of lectures
Introduction to Java	1-8	6
Objects and Classes	9	3
Strings	4.4, 10.10, 10.11	2
Thinking in Objects	10	2
Inheritance and Polymorphism	11	3
Midterm Exam (30%)		
Abstract Classes and Interfaces	13	3
Exception Handling and Text I/O	12	3
JavaFX Basics	14	3
JavaFX UI Controls	16	3
Event-Driven Programming	15	3
Final Exam (35%)		

Lab Outline

Lab #	Title	Quizzes
1	Program structure in Java	
2	Structure Programming - Revision	
3	Methods	
4	Arrays and Object Use	Q1
5	Object-Oriented Programming	
6	String I	
7	String II	Q2
8	Inheritance and Polymorphism	
9	Abstract classes and Interfaces	
10	Text I/O	Q3
11	GUI	
12	Event-Driven Programming	Q4
Practical Final Exam (10%)		

Why Java?

❖There are many PLs: C#, PhP, Python, C++, VB.NET.

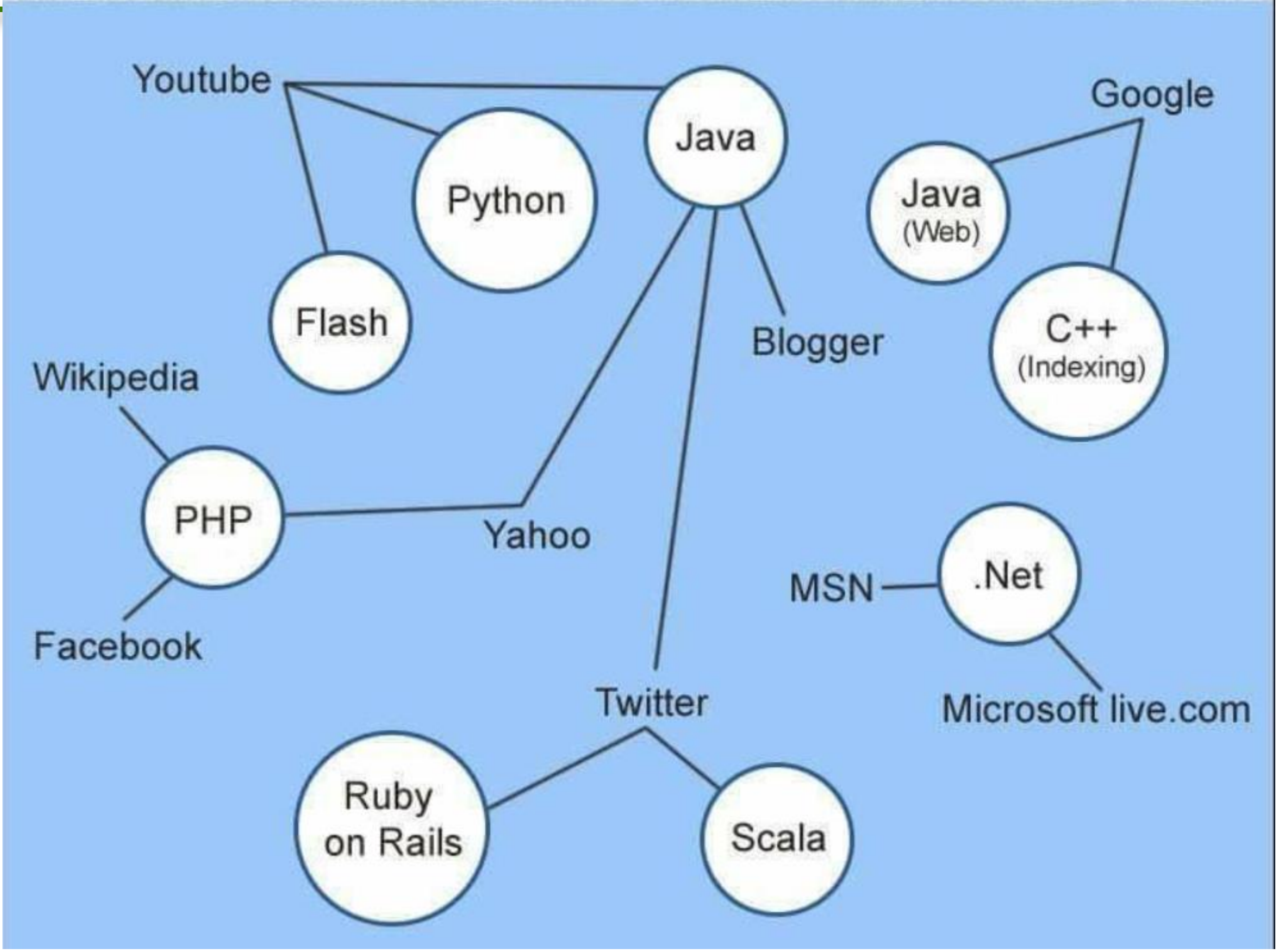
❖There is no best PL. Each has its own purpose.

❖Java is a general purpose programming language.

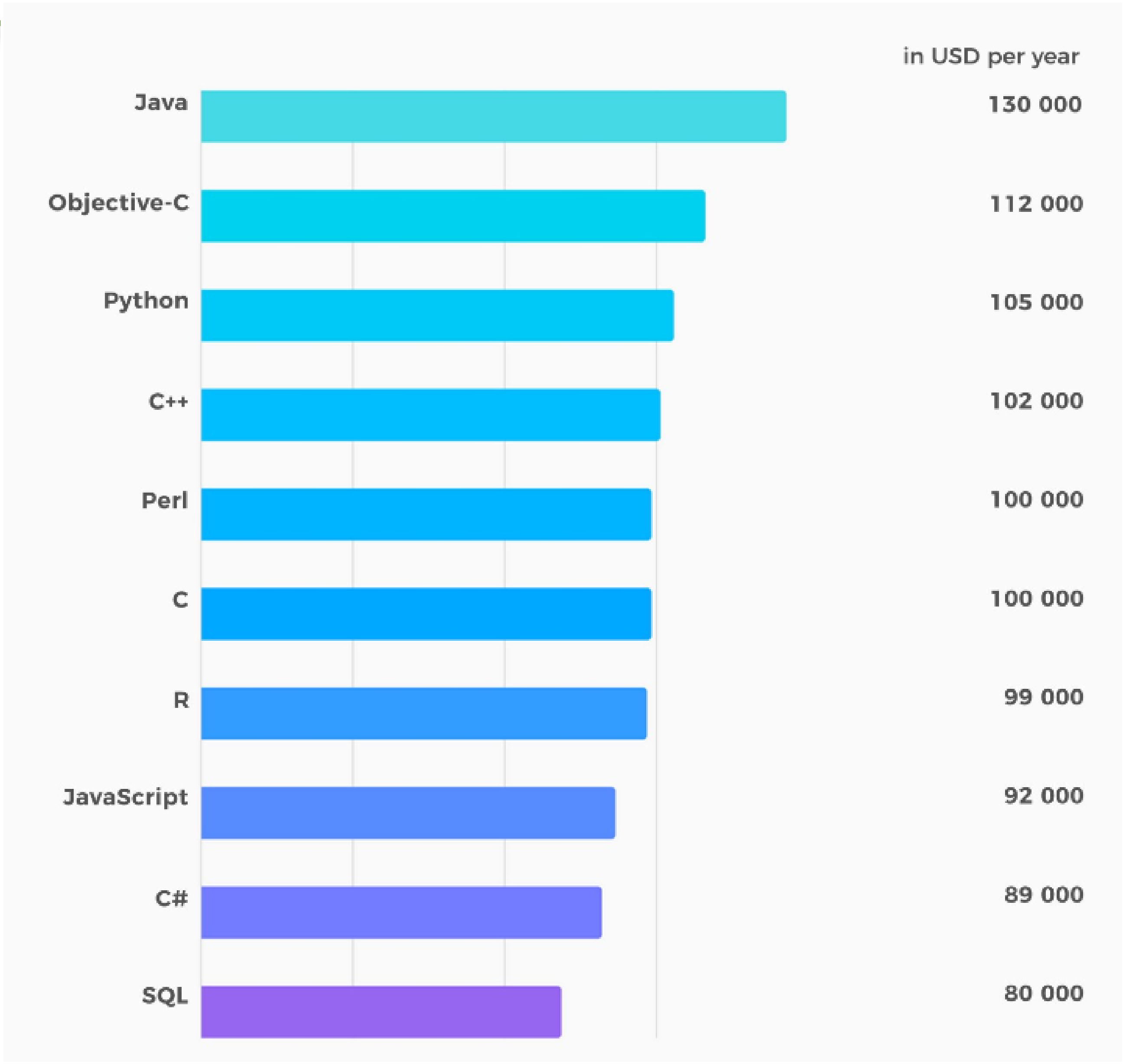
❖ Java is the Internet programming language.

❖Java is Mission-Critical Choice

LANGUAGES USED



$Average Salaries in the US



Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

JDK Versions

- JDK 1.02 (1995)
- JDK 1.1 (1996)
- JDK 1.2 (1998)
- JDK 1.3 (2000)
- JDK 1.4 (2002)
- JDK 1.5 (2004) a. k. a. JDK 5 or Java 5
- JDK 1.6 (2006) a. k. a. JDK 6 or Java 6
- JDK 1.7 (2011) a. k. a. JDK 7 or Java 7
- **JDK 8 (Update 72 January 2016)**

JDK Editions

- **Java Standard Edition (J2SE)**
  - J2SE can be used to develop client-side standalone applications or applets.
- **Java Enterprise Edition (J2EE)**
  - J2EE can be used to develop server-side applications such as Java servlets, Java ServerPages, and Java ServerFaces.
- **Java Micro Edition (J2ME).**
  - J2ME can be used to develop applications for mobile devices such as cell phones.

Popular Java IDEs

IDE → Integrated **D**evelopment **E**nvironment

eclipse

NetBeans

IJ

A Simple Java Program

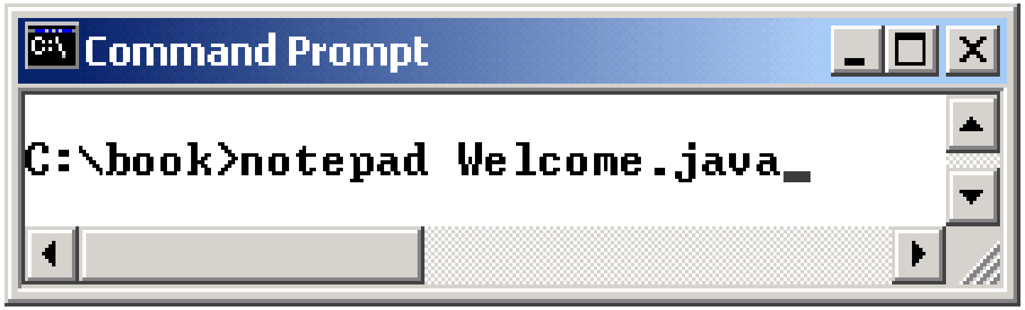
```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Creating and Editing Using NotePad

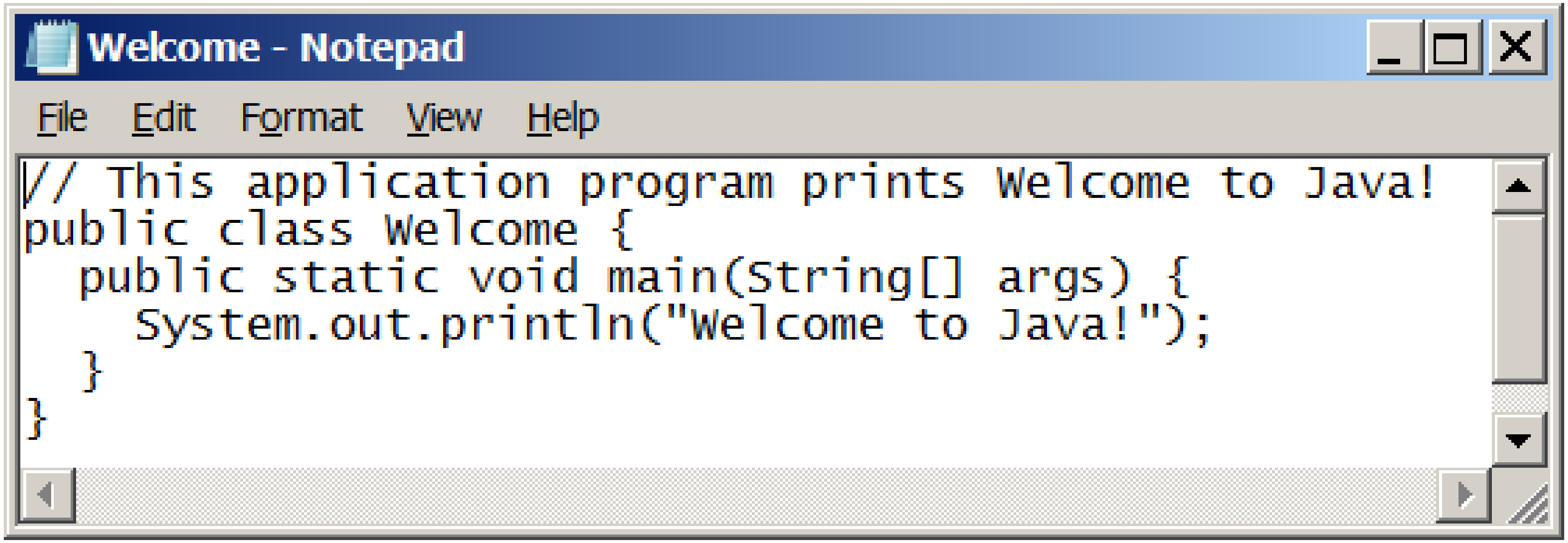
To use NotePad, type:

notepad Welcome.java

from the **DOS** prompt.

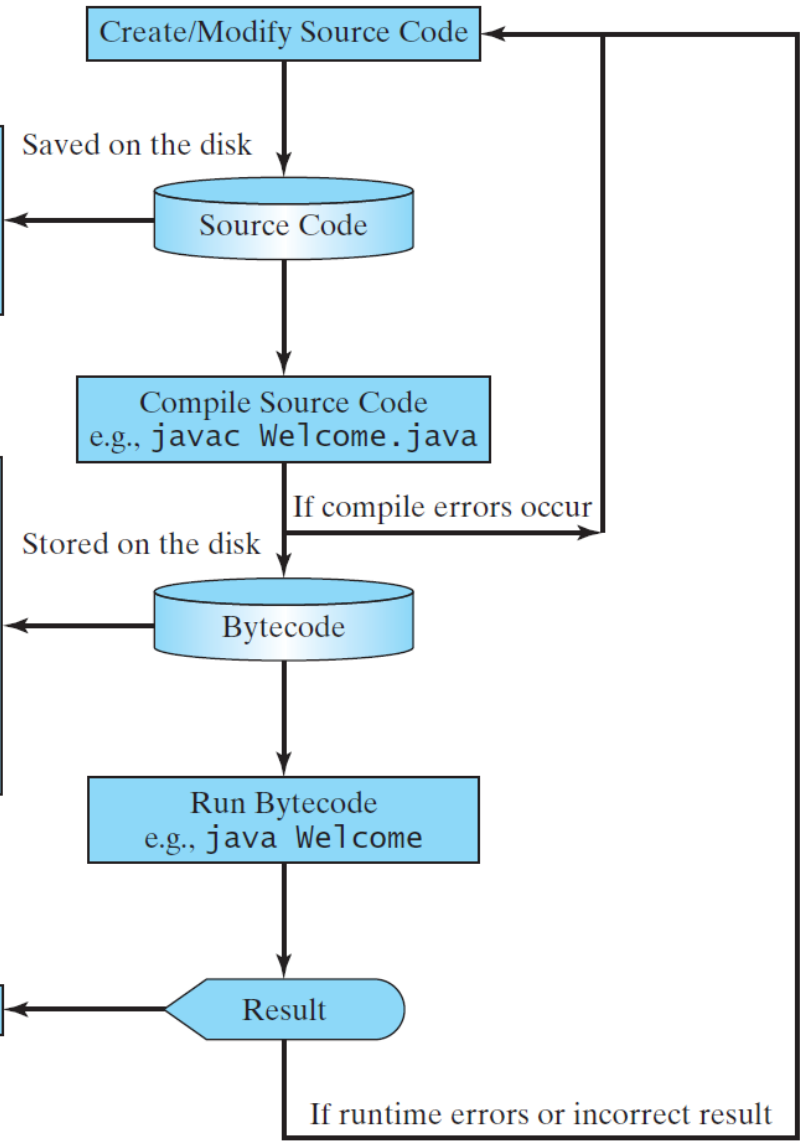


```
C:\book>notepad Welcome.java
```



```
// This application program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Creating, Compiling, and Running Programs



Source code (developed by the programmer)

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

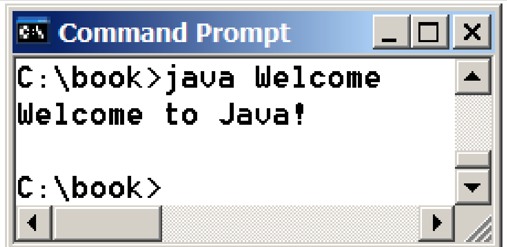
Bytecode (generated by the compiler for JVM to read and interpret)

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

"Welcome to Java " is displayed on the console

```
Welcome to Java!
```



```
C:\book>java Welcome
Welcome to Java!

C:\book>
```

Compiling and Running Java from the Command Window

- Set path to JDK **bin** directory

```
set path=c:\Program Files\java\jdk1.8.0_xx\bin
```

- Set **classpath** to include the current directory

```
set classpath=.
```

- Compile:

```
javac Welcome.java
```

- Run:

```
java Welcome
```

Anatomy of a Java Program

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks

Class Name

❖ Every Java program must have **at least** one class.

❖ Each class has a name.

❖ By **convention**, class names start with an uppercase letter.

❖ In this example, the class name is **Welcome**.

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
        System.out.println("Welcome to Java!");
  }
}
```

Main Method

❖ In order **to run a class**, the class must contain a method named **main**.

❖ The program is executed from the **main** method.

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Statement

❖ A statement represents an action or a sequence of actions.

❖ The statement **System.out.println("Welcome to Java!")** in the program is a statement to display the greeting "*Welcome to Java*!".

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
        System.out.println("Welcome to Java!");
  }
}
```

Statement Terminator

- **Every** statement in Java ends with a semicolon

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
      System.out.println("Welcome to Java!");
  }
}
```

Reserved Words

❖ Reserved words or **keywords** are words that have a specific meaning to the compiler and cannot be used for other purposes in the program.

❖ For example, when the compiler sees the word **class**, it understands that the word after class is the name for the class.

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
      System.out.println("Welcome to Java!");
  }
}
```

Programming Style and Documentation

- ❖ Appropriate **Comments**.
- ❖ Naming **Conventions**.
- ❖ Proper **Indentation** and Spacing Lines.
- ❖ Block Styles.

Naming Conventions

- Choose **meaningful** and descriptive names.
- Class names:
  - Capitalize the **F**irst **L**etter of each word in the name. For example, the class name **ComputeExpression**.

Proper Indentation and Spacing

- **Indentation**
  - Indent **two** spaces.
- **Spacing**
  - Use blank line to separate segments of the code.

Block Styles

Next-line style

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Programming Errors

- **Syntax Errors**
  - Detected by the compiler
- **Runtime Errors**
  - Causes the program to abort
- **Logic Errors**
  - Produces incorrect result