BIRZEIT UNIVERSITY

COMP1331

COMPUTER & PROGRAMMING

By: Mamoun Nawahdah (Ph.D.)
2022

Welcome to COMP1331, one of the most **exciting** programming courses offered at Computer Science Department

Course Description

- Programming in one of high level languages: **Java**
- Basic structures of programming tools: language elements, control statements, methods, arrays, strings, file processing, objects and classes, thinking in objects, and introduction to inheritance and polymorphism

Logistics

- Instructor: **Mamoun Nawahdah** (**WKS205**)
- Text book:
  - Introduction To JAVA Programming, **12th** edition.
  - Author: Y. Daniel Liang.
  - Publisher: Prentice Hall.
- Lab Manual:
  - **Title:** COMP1331 Lab Work

Special Regulations

- **Multi-Phase Project**:
  - 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 ITC** 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/lab. If your mobile ring you have to leave the classroom **q**uickly, **q**uietly and don't come back.
- **Late**: you are expected to be in the classroom/lab before the teacher arrival. After **5** minutes you will not allowed entering the classroom/lab.

Grading Criteria

Midterm exam	30%
Multi-phase project + discussion	15%
Quizzes	10%
Final Practical Exam	10%
Final exam	35%

Course Outline

Topics	Chapter	# of lectures
Introduction to Java	1-6	8
Recursion	18	2
Objects and Classes	9	5
Arrays	7,8	4
Midterm Exam (30%)		
Strings	10	3
Introduction to Exception Handling and Text I/O	12	3
Object-Oriented Thinking	10	3
Introduction to Inheritance and Polymorphism	11	2
Total # of Lectures		30
Final Exam (35%)		

Lab Outline

Lab #	Title	Quizzes
1	Elementary Java Programming	
2	Selections	
3	Loops	
4	Methods	Q1 (Lab1,2,3)
5	Recursion	
6	Objects and Classes 1	
7	Objects and Classes 2	Q2 (Lab4,5,6)
8	Single-Dimensional Arrays	
9	Multidimensional Arrays	
10	Strings	Q3 (Lab7,8,9)
11	Text I/O	
12	Class Relationships	Q4 (Lab10,11)
Practical Final Exam (10%) (Lab 1 to 12)		

What is a Computer?

Programs

Computer *programs*, known as ***software***, are instructions to the computer.

You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.

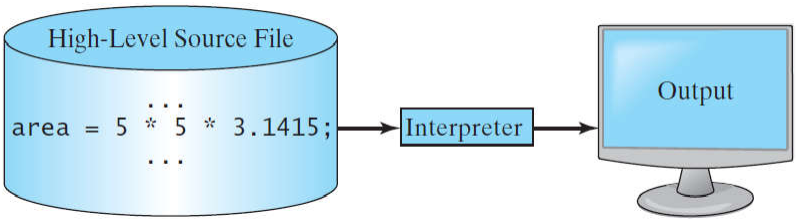
Programs are written using programming languages.

Interpreting/Compiling Source Code

- A program written in a **high-level language** is called a ***source program*** or ***source code***.
- Because a computer cannot understand a source program, a source program must be translated into **machine code** for execution.
- The translation can be done using another programming tool called an ***interpreter*** or a ***compiler***.

Interpreting Source Code

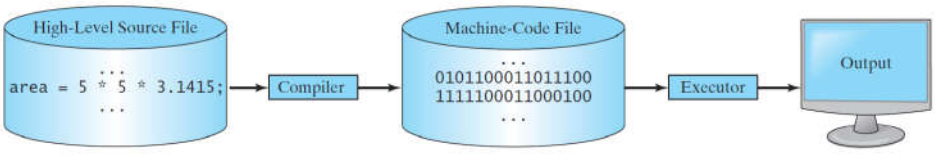
- An interpreter reads one statement from the source code, translates it to the machine code or **virtual machine code**, and then executes it right away, as shown in the following figure.



- Note that a statement from the source code may be translated into several machine instructions.

Compiling Source Code

- A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in the following figure.



Why Java?

- Java is a general purpose programming language.
- Java is the Internet programming language.

James Gosling, the creator of Java

Java, Web, and Beyond

- Java can be used to develop standalone applications.
- Java can be used to develop applications running from a browser.
- Java can also be used to develop applications for hand-held devices.
- Java can be used to develop applications for Web servers.

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

17

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) JDK 5
- JDK 1.6 (2006) JDK 6
- JDK 1.7 (2011) JDK 7

- **JDK 8 (2014)**
- **JDK 10 (March 2018)**
- **JDK 11 (September 2018)**
- **JDK 12 (March 2019)**
- **JDK 13 (January 2020)**
- **JDK 14 (July 2020)**
- **JDK 16 (March 2021)**
- **JDK 17 (Oct 2021)**

18

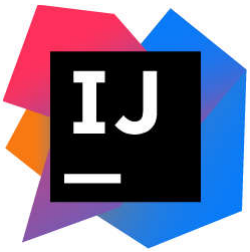
JDK Editions

- **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.

19

Popular Java IDEs

IDE ➔ **I**ntegrated **D**evelopment **E**nvironment



20

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!");
  }
}
```

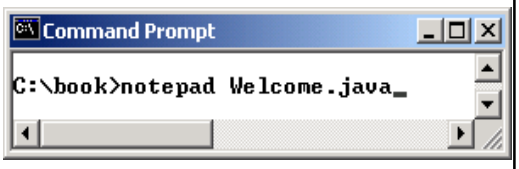
21

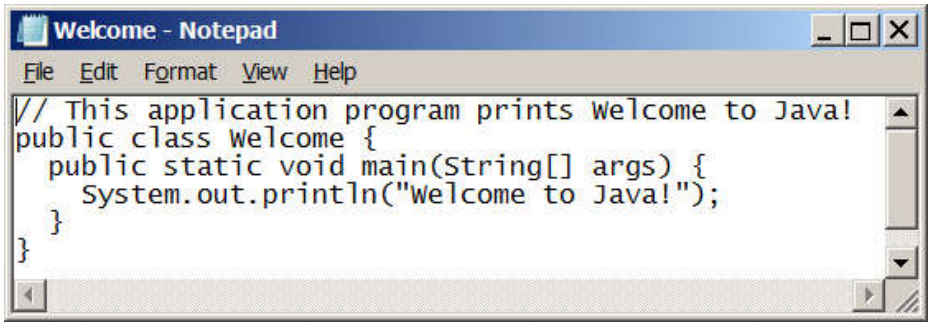
Creating and Editing Using NotePad

To use NotePad, type:

notepad Welcome.java

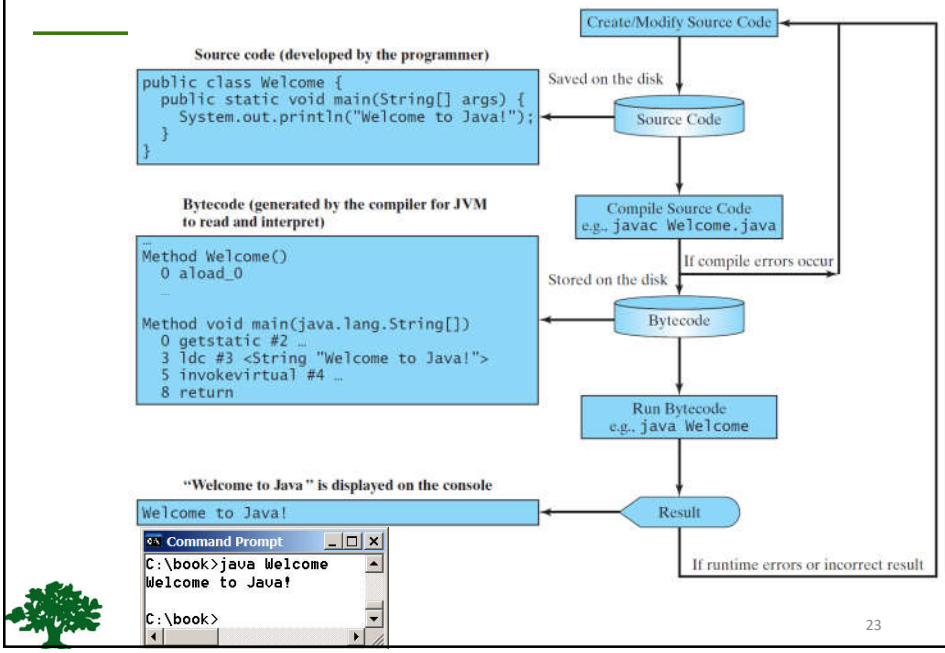
from the **DOS** prompt.





22

Creating, Compiling, and Running Programs



23

Compiling and Running Java from the Command Window (cmd)

- Set path to JDK **bin** directory

  set path=C:\Program Files\Java\jdk-17\bin

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

  set classpath=.

- Compile:

  javac Welcome.java

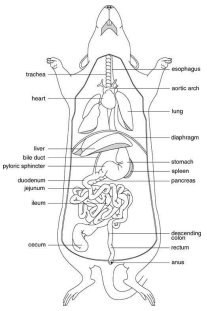
- Run:

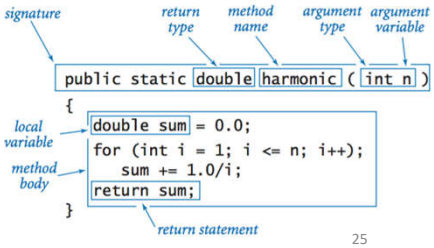
  java Welcome

24

Anatomy of a Java Program

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





25

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!");
  }
}
```

26

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!");
  }
}
```

27

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*!**".

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

28

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!");
  }
}
```

29

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!");
  }
}
```

30

Programming Style and Documentation

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

31

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**.

32

Proper Indentation and Spacing

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

33

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");
  }
}
```

34

Programming Errors

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

35

Syntax Errors

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

36

Runtime Errors

```
public class ShowRuntimeErrors {
  public static void main(String[] args)
  {
    System.out.println(1 / 0);
  }
}
```

37

Logic Errors

```
public class ShowLogicErrors {
  public static void main(String[] args) {
    System.out.println("Celsius 35 is Fahrenheit
  degree ");
    System.out.println((9 / 5) * 35 + 32);
  }
}
```

38