1

Introduction to OOP Using Java

Introduction

- **Sun's implementation called the Java Development Kit (JDK)**
- **Object-Oriented Programming**
- **Java is language of choice for networked applications**
- **Java Enterprise Edition (Java EE) geared toward large-scale distributed applications and web applications**
- **Java Micro Edition (Java ME) geared toward applications for small, memory constrained devices**

Machine Languages, Assembly Languages and High-Level Languages

- **Machine language**
  - **"Natural language" of computer component**
  - **Machine dependent**
- **Assembly language**
  - **English-like abbreviations represent computer operations**
  - **Translator programs (assemblers) convert to machine language**
- **High-level language**
  - **Allows for writing more "English-like" instructions**
    - **Contains commonly used mathematical operations**
  - **Compiler converts to machine language**
- **Interpreter**
  - **Execute high-level language programs without compilation**

History of C and C++

- C++ evolved from C, which evolved from BCPL and B
- C
  - Developed at Bell Labs
  - Popularized as the language of the UNIX operating system
- C++
  - Deveoped by Bjarne Stroustrup
  - Provides object-oriented programming capabilities
  - Hybrid language

History of Java

- **Java**
  - **Originally for intelligent consumer-electronic devices**
  - **Then used for creating web pages with dynamic content**
  - **Now also used to:**
    - **Develop large-scale enterprise applications**
    - **Enhance web server functionality**
    - **Provide applications for consumer devices (cell phones, etc.)**

Java Class Libraries

- Java programs consist of classes
  - Include methods that perform tasks
    - Return information after task completion
- Java provides class libraries
  - Known as Java APIs (Application Programming Interfaces)
- To use Java effectively, you must know
  - Java programming language
  - Extensive class libraries

Software Engineering Observation

When programming in Java, you will typically use the following building blocks: Classes and methods from class libraries, classes and methods you create yourself and classes and methods that others create and make available to you.

Performance Tip

Using Java API classes and methods instead of writing your own versions can improve program performance, because they are carefully written to perform efficiently. This technique also shortens program development time.

Portability Tip

Using classes and methods from the Java API instead of writing your own improves program portability, because they are included in every Java implementation.

Typical Java Development Environment

- **Java programs go through five phases**
  - **Edit**
    - **Programmer writes program using an editor; stores program on disk with the `.java` file name extension**
  - **Compile**
    - **Use javac (the Java compiler) to create bytecodes from source code program; bytecodes stored in `.class` files**
  - **Load**
    - **Class loader reads bytecodes from `.class` files into memory**
  - **Verify**
    - **Bytecode verifier examines bytecodes to ensure that they are valid and do not violate security restrictions**
  - **Execute**
    - **Java Virtual Machine (JVM) uses a combination of interpretation and just-in-time compilation to translate bytecodes into machine language**

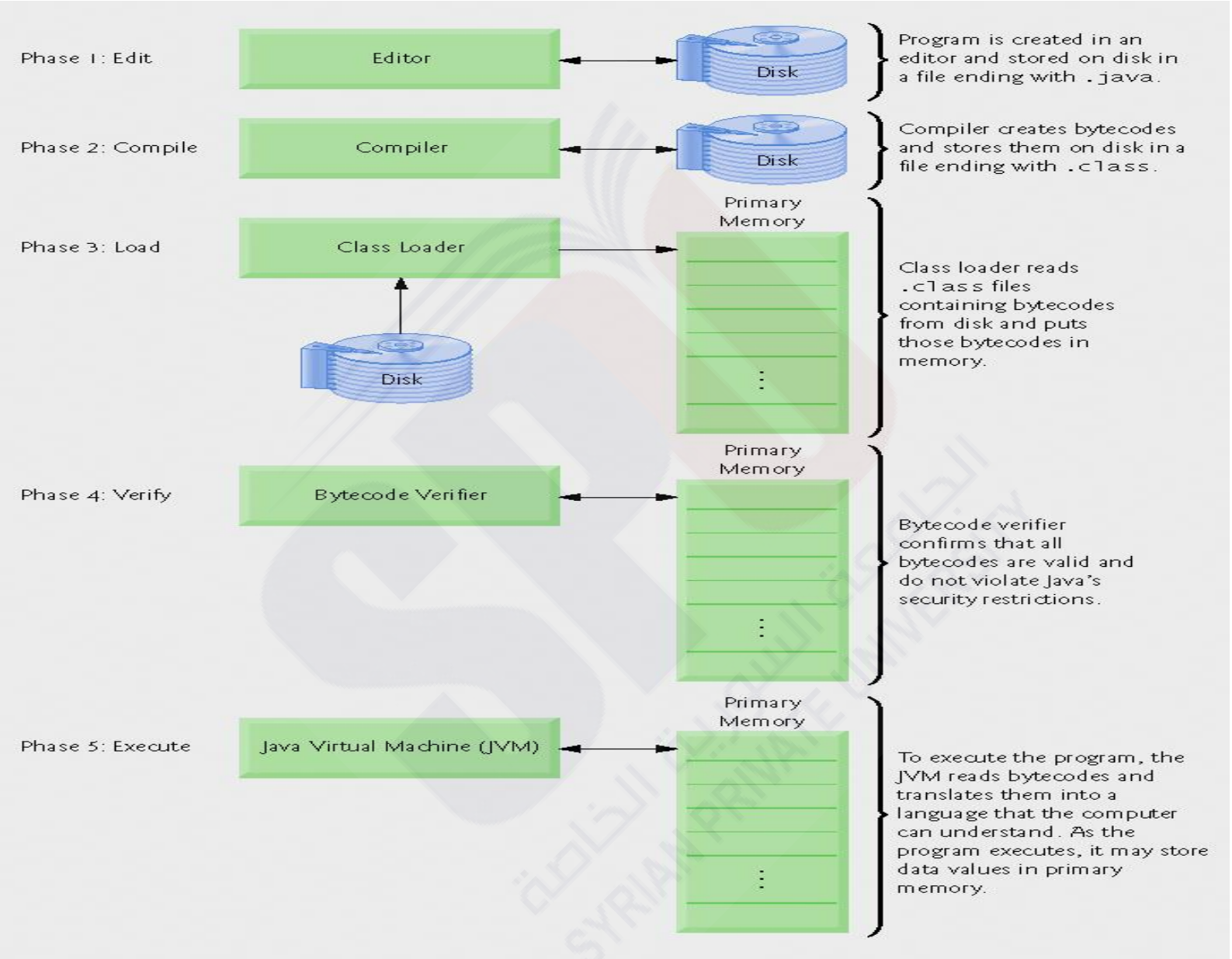


Fig. 1| Typical Java development environment.

What is Object?

- **Objects:**
  - **Reusable software components that model real-world items**
  - **Look all around you**
    - **People, animals, plants, cars, etc.**
  - **Attributes**
    - **Size, shape, color, weight, etc.**
  - **Behaviors**
    - **Babies cry, crawl, sleep, etc.**

What is OOP?

- **Object-oriented design (OOD)**
  - **Models software in terms similar to those used to describe real-world objects**
  - **Class relationships**
  - **Inheritance relationships**
  - **Models communication among objects**
  - **Encapsulates attributes and operations (behaviors)**
    - **Information hiding**
    - **Communication through well-defined interfaces**
- **Object-oriented language**
  - **Programming in object-oriented languages is called *object-oriented programming* (*OOP*)**
  - **Classes are to objects as blueprints are to houses**

First Program in Java: Printing a Line of Text

```
1  // Fig. 2.1: Welcome1.java
2  // Text-printing program.
3
4  public class Welcome1
5  {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10
11    } // end method main
12
13 } // end clazss Welcome1
```

```
Welcome to Java Programming!
```

First Program in Java: Printing a Line of Text (Cont.)

```
1  // Fig. 2.1: Welcome1.java
```

- Comments start with: //
  - Comments ignored during program execution
  - Document and describe code
  - Provides code readability
- Traditional comments: /* ... */

```
/* This is a traditional
   comment. It can be
   split over many lines */
```

```
2  // Text-printing program.
```

- Another line of comments
- Note: line numbers not part of program, added for reference

First Program in Java: Printing a Line of Text (Cont.)

```
3
```

- **Blank line**
  - **Makes program more readable**
  - **Blank lines, spaces, and tabs are white-space characters**
    - **Ignored by compiler**

```
4   public class Welcome1
```

- **Begins class declaration for class `Welcome1`**
  - **Every Java program has at least one user-defined class**
  - **Keyword: words reserved for use by Java**
    - **`class` keyword followed by class name**
  - **Naming classes: capitalize every word**
    - **`SampleClassName`**

First Program in Java: Printing a Line of Text (Cont.)

```
4   public class Welcome1
```

- Java identifier
  - Series of characters consisting of letters, digits, underscores (_) and dollar signs ($)
  - Does not begin with a digit, has no spaces
  - Examples: `Welcome1`, `$value`, `_value`, `button7`
    - `7button` is invalid
  - Java is case sensitive (capitalization matters)
    - `a1` and `A1` are different

Good Programming Practice

- **By convention, always begin a class name's identifier with a capital letter and start each subsequent word in the identifier with a capital letter. Java programmers know that such identifiers normally represent Java classes, so naming your classes in this manner makes your programs more readable.**

First Program in Java: Printing a Line of Text (Cont.)

```
7      public static void main( String args[] )
```

- Part of every Java application
  - Applications begin executing at `main`
    - Parentheses indicate `main` is a method
    - Java applications contain one or more methods
  - Exactly one method must be called `main`
- Methods can perform tasks and return information
  - `void` means `main` returns no information
  - For now, mimic `main`'s first line

```
8      {
```

- Left brace begins body of method declaration
  - Ended by right brace } (line 11)

First Program in Java: Printing a Line of Text (Cont.)

```
9         System.out.println( "Welcome to Java Programming!" );
```

- Instructs computer to perform an action
  - Prints string of characters
    - String – series of characters inside double quotes
  - White-spaces in strings are not ignored by compiler
- `System.out`
  - Standard output object
- Method `System.out.println`
  - Displays line of text
- This line known as a statement
  - Statements must end with semicolon `;`

First Program in Java: Printing a Line of Text (Cont.)

```
11    } // end method main
```

- Ends method declaration

```
13 } // end class Welcome1
```

- Ends class declaration
- Can add comments to keep track of ending braces

Outline

```
1  // Fig. 2.4: Welcome3.java
2  // Printing multiple lines of text with a single statement.
3
4  public class Welcome3
5  {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9        System.out.println( "Welcome\nto\nJava\nProgramming!" );
10
11    } // end method main
12
13 } // end class Welcome3
```

```
Welcome
to
Java
Programming!
```

A new line begins after each \n escape sequence is output.

- Welcome3.java

  1. main

  2. System.out.println
  (uses \n **for new line)**

- **Program Output**

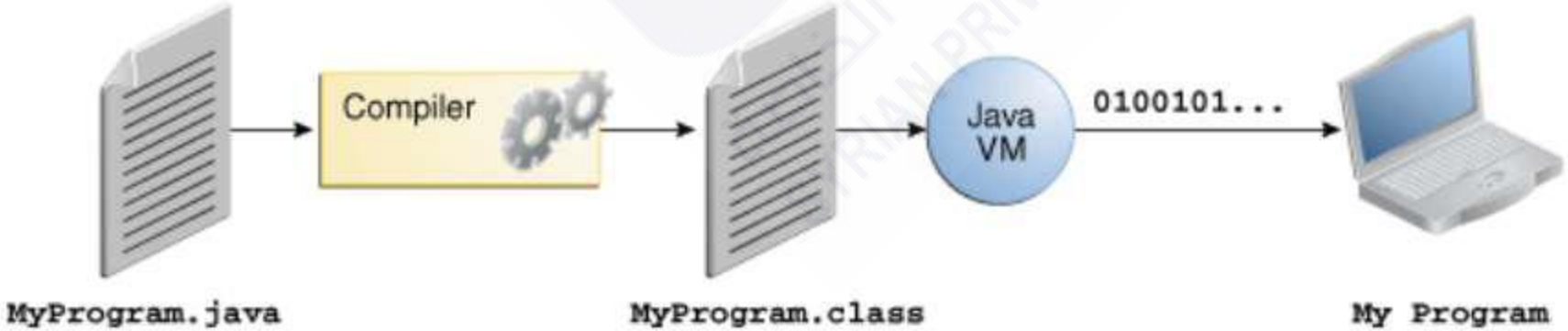
Escape sequence	Description
`\n`	Newline. Position the screen cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor at the beginning of the current line—do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
`\\`	Backslash. Used to print a backslash character.

`\"`	Double quote. Used to print a double-quote character. For example,
	`System.out.println( "\"in quotes\"" );`
	displays
	`"in quotes"`

Introduction to Java technology

The **Java programming language** is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Portable

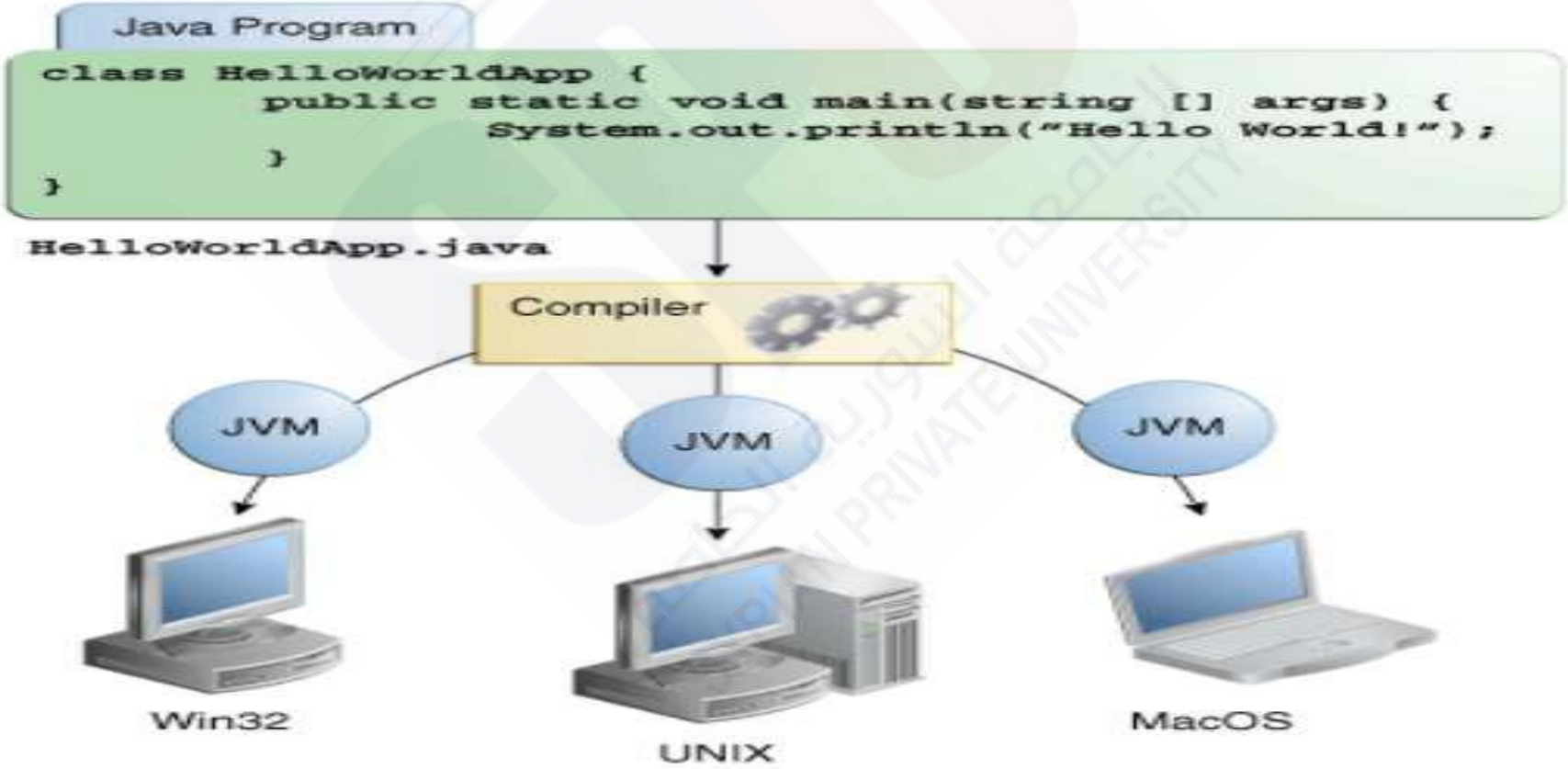


Why Java is OOP?

- **Encapsulation--**implements information hiding and modularity (abstraction)
- **Polymorphism--**the same message sent to different objects results in behavior that's dependent on the nature of the object receiving the message
- **Inheritance--**you define new classes and behavior based on existing classes to obtain code re-use and code organization

Why Java is considered portable?

Portable: The primary benefit of the interpreted byte code approach is that compiled Java language programs are portable to any system on which the Java interpreter and run-time system have been implemented.



Typical Java Development Environment

Integrated development environments (IDEs): Provide tools that support the software-development process, including editors for writing and editing programs and debuggers for locating logic errors—errors that cause programs to execute incorrectly.

Popular IDEs

- **Eclipse (**www.eclipse.org**)**
- **NetBeans (**www.netbeans.org**)**
- **JBuilder (**www.codegear.com**)**
- **JCreator (**www.jcreator.com**)**
- **BlueJ (**www.blueJ.org**)**
- **jGRASP (**www.jgrasp.org**)**