Structured Programming

Lecture 1

Dr. Obead Alhadreti

Outline

- Course Overview
- Programming Languages
- Java Programming Language
- Classes & Objects: Defining, Creating, and Using.

Course Overview

- **Timetable**

  A 2-hour lecture + A 1-hour lab. + A 2-hour lab.

- **Course Learning Objectives:**

1. Students will understand object oriented concepts including classes, objects, inheritance, data abstraction, encapsulation, and polymorphism;
2. Students will learn how to design applications using object oriented design methodology;
3. Students will appreciate the benefits of code reuse by learning how to make use of off-the-shelf Java libraries.

Course Overview

- **Course Elements**

1. Lectures

Can be downloaded from your the portal of "Learning"

2. Lab Exercises

3. Assignments

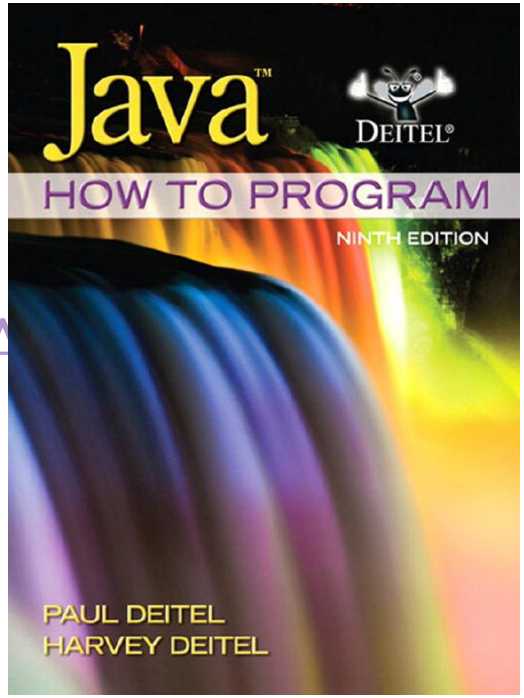
4. Exams

Course Overview

Topic to be covered:
1. Overview of Object Oriented Programming
2. Classes and Objects
3. Modifiers
4. Encapsulation
5. Inheritance
6. Overriding and Overloading
7. Interfaces
8. Abstraction
9. Polymorphism
10. UML for Object Oriented Programming

Course Overview

- **Textbook:**
  - Java: How to Program, 9th edition , Dietel and Dietel, Pearson 0273759760.
  - Available at the college library.
  - Parts available @ http://www.deitel.com/Books/Java/JavaHowyObjectsVersion/tabid/3622/Default.aspx



Additional Resources

- Harmash.com (Arabic)

http://www.harmash.com/java/lesson_1.html

- Rwaq (Arabic)

https://www.rwaq.org/

- w3resource (English)

https://www.w3resource.com/

Course Overview

- **Assessment Methods:**

- Final Paper-based Exam (40%)
- Final Lab Exam (10%)
- Mid-term Exam (20%)
- Assignments (20%)
- Lab exercises (10%)

Course Overview

- Students are expected to attend lectures in time.
- Attendance is monitored by means of an attendance sheet. This is filled in within 10 minutes of the start of each lecture.
- Students are responsible to submit assignments in time.
- Late submissions will be subject to penalties.
- There will be no makeup exams except under emergencies. If a student cannot attend the exam, then student must make arrangement with the instructor prior to the planned absence.

Programming Languages

Programming Languages

- For a computer to be able to perform specific tasks (i.e. print what grade a student got on an exam), it must be given instructions to do the task.

- The set of instructions that tells the computer to perform specific tasks is known as software program.

- A *programming language* is a set of rules, symbols and special words used to write statements in order to develop sets of instructions (computer software) for computers to execute.

Examples of Programming Languages

ActionScript
ALGOL
Ada
AIML
Assembly
AutoHotkey
Babel
BASIC
Batchfile
BCPL
Brooks
C
C#
C++
Clojure
COBOL
CoffeeScript
CPL
Curl
Curry

D
DarkBASIC
Datalog
dBASE
Dylan
F
F#
FORTRAN
FoxPro
Go
GW Basic
Haskell
HDML
HTML
Java
JavaScript
JCL
Julia

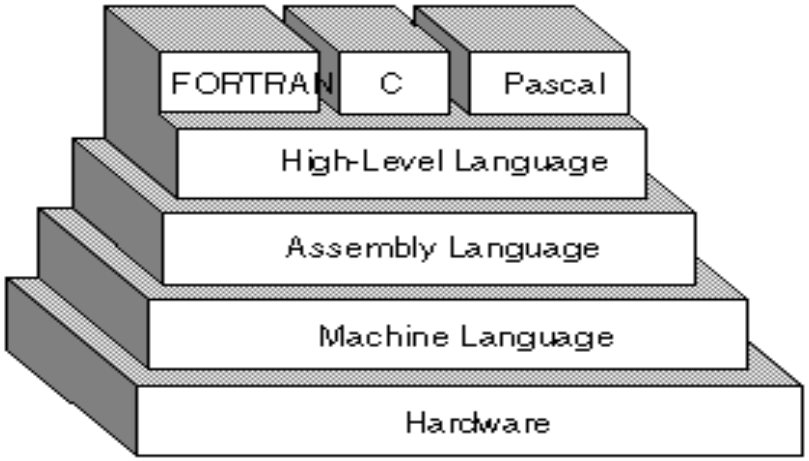
LISP
Live Script
LOGO
Lua
Matlab
MUMPS
Nim
Objective-C
OCaml
Pascal
Perl
PHP
Pick
PureBasic
Python
Prolog
QBasic

R
Racket
Reia
Ruby
Rust
Scala
Scheme
SGML
Simula
Smalltalk
SQL
Tcl
Turbo Pascal
True BASIC
VHDL
Visual Basic
Visual FoxPro
WML
WHTML
XML

Programming Language Levels

- There are two main levels of programming languages.

1. Low-level programming. Low-level languages are closer to the hardware than are high-level programming. Low-level languages can convert to machine code without a compiler or interpreter. For example, assembly language.



Programming Language Levels

2. high-level programming languages: languages that are closer to human languages and further from machine languages. Examples include Java, C/C++ , Pascal, FORTRAN , BASIC

- The main advantage of high-level languages over low-level languages is that they are easier to read, write, and maintain.

- Programs written in a high-level language must be *translated* into machine language by a compiler or interpreter.

Programming Paradigms

- A programming paradigm is a model of programming based on distinct concepts that shapes the way programmers design, organize and write programs.

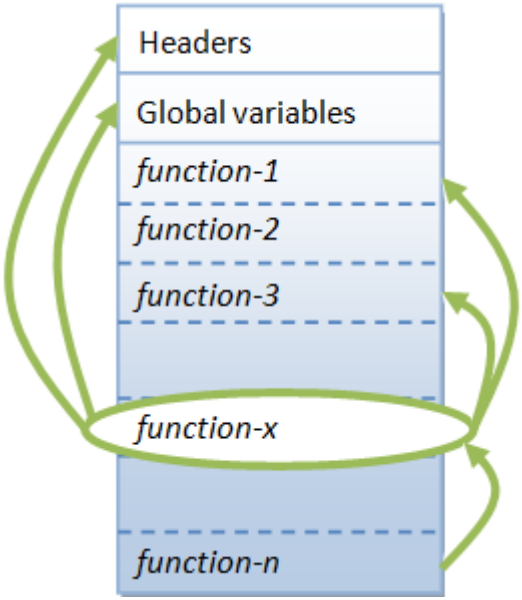
- There are two main programming paradigms:

1. Procedural programming

2. Object-oriented paradigm

1. Procedural Programming

- In procedural (functions-based) programming, programming relies on procedures (functions) which contain a series of computational steps to be carried out. Code is executed from the top of the file to the bottom.



1. Procedural Programming

- The life of a process-centred design was short because changes to the process specification required a *change* in the *entire program*. An inability to reuse existing code without considerable overhead.

- Examples of computer procedural languages are BASIC, FORTRAN, and Pascal.

2. Object-Oriented Programming (OOP)

- To have a fine definition of OOP, please note what do you see in your classroom right now? Nice, what is the attributes and behavior of each of them ?

- OOP program is a collection of interacting objects. That's mean objects play a *central* role.

- **Each** object consists of attributes and behavior.

- **Each** different type of object comes from a specific class of that type.

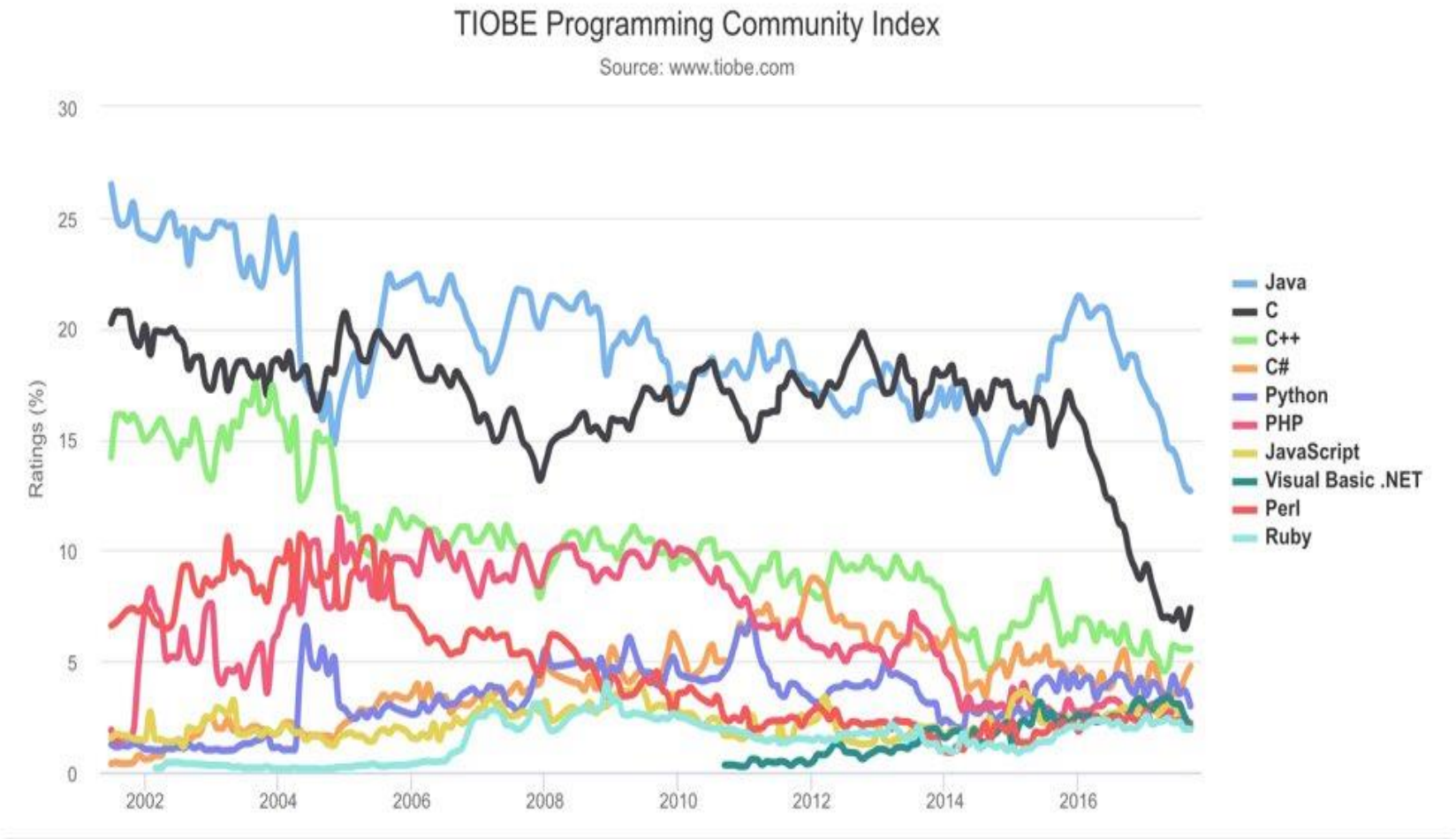
Some Benefits of OOP

- OOP is easier to accommodate changes.
- It helps increase productivity through reuse of existing codes.
- The OOP paradigm is currently the most popular way of analysing, designing, and developing application systems, especially **large ones**.
- Examples of OOP languages include: Java, Python, and C++.

Java Programming Language

What is Java ?

- It is a high-level programming language
- It was originally developed by **Sun Microsystems in** 1995, and now owned by **Oracle Corporation**.
- One of the best programming language in the last 20 years.


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
Java
C
C++
C#
Python
PHP
JavaScript
Visual Basic .NET
Perl
Ruby

Reasons to Learn Java

1. Java is an OOP Language
2. Java is easy to learn
3. Java is FREE
4. Wonderful community support
5. Great collection of Open Source libraries
6. Excellent documentation support
7. Java is everywhere: Desktop, web applications, and mobile applications.
8. Java support *Platform Independent Compiling*

Programming Language Compiler

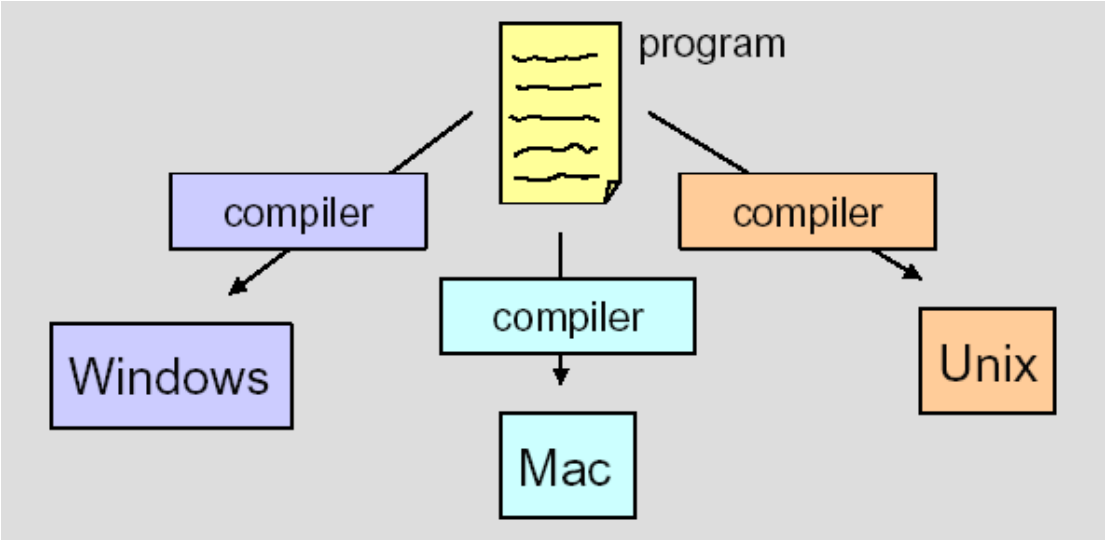
- A compiler is a software that:

1. *Checks the correctness* of the source code according to the language rules.
   - **Syntax errors** are raised if some rules were violated.

2. *Translates* the source code into a machine code if no errors were found.

Platform dependent Compiling

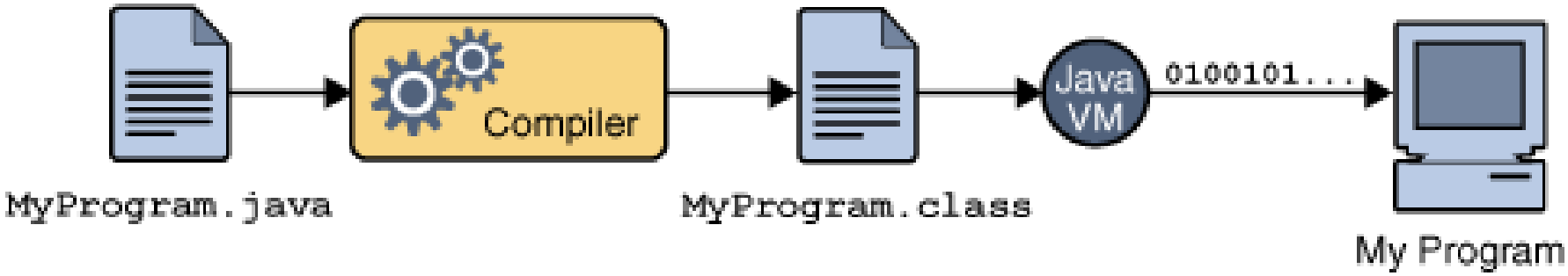
- Because different operating systems (windows, macs, unix), require different machine code, you must compile most programs separately for each platform.



Java Platform Independent Compiling

- The Java **c**ompiler (**javac**) produces *bytecode (a ".class" file)* not machine code from the source code (the ".java" file).

- Java Virtual Machine (JVM): A hypothetical computer developed to make Java programs machine independent (i.e run on many different types of computer platforms). Bytecode is the machine language for the JVM

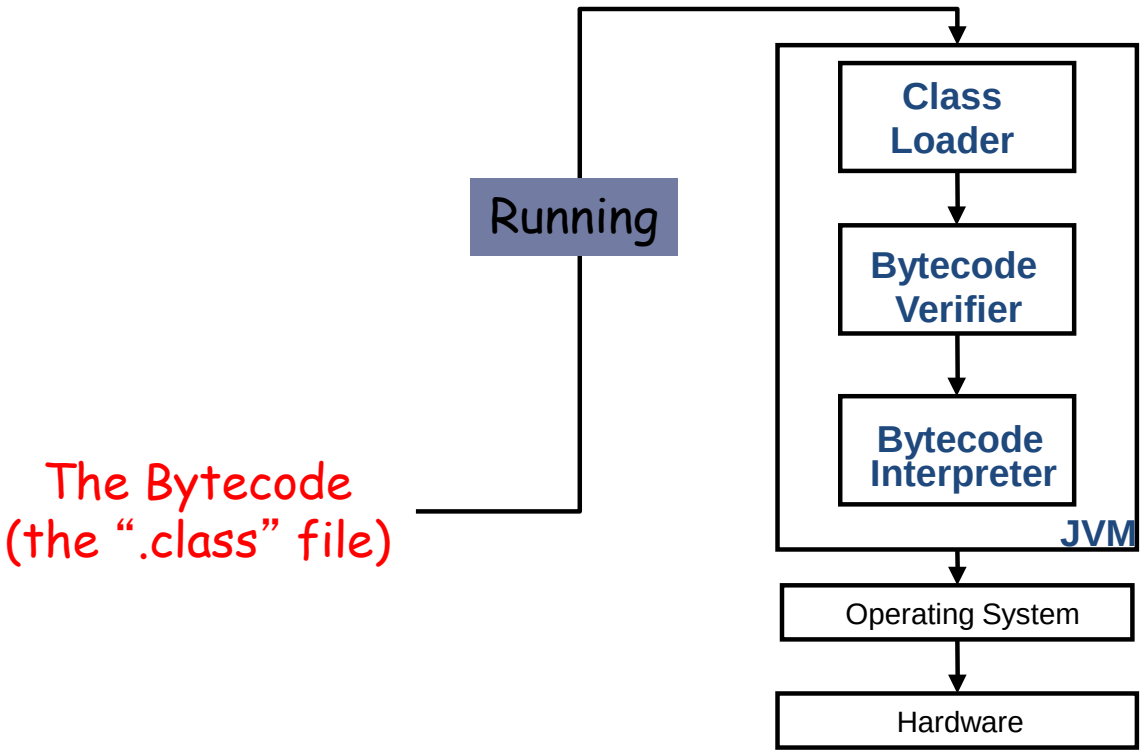


Java Platform Independent Compiling

- **The Java Virtual Machine (JVM) Components:**

1. **The Class Loader**
   - stores bytecodes in memory
2. **Bytecode Verifier**
   - ensures bytecodes do not violate security requirements
3. **Bytecode Interpreter**
   - translates bytecodes into machine language

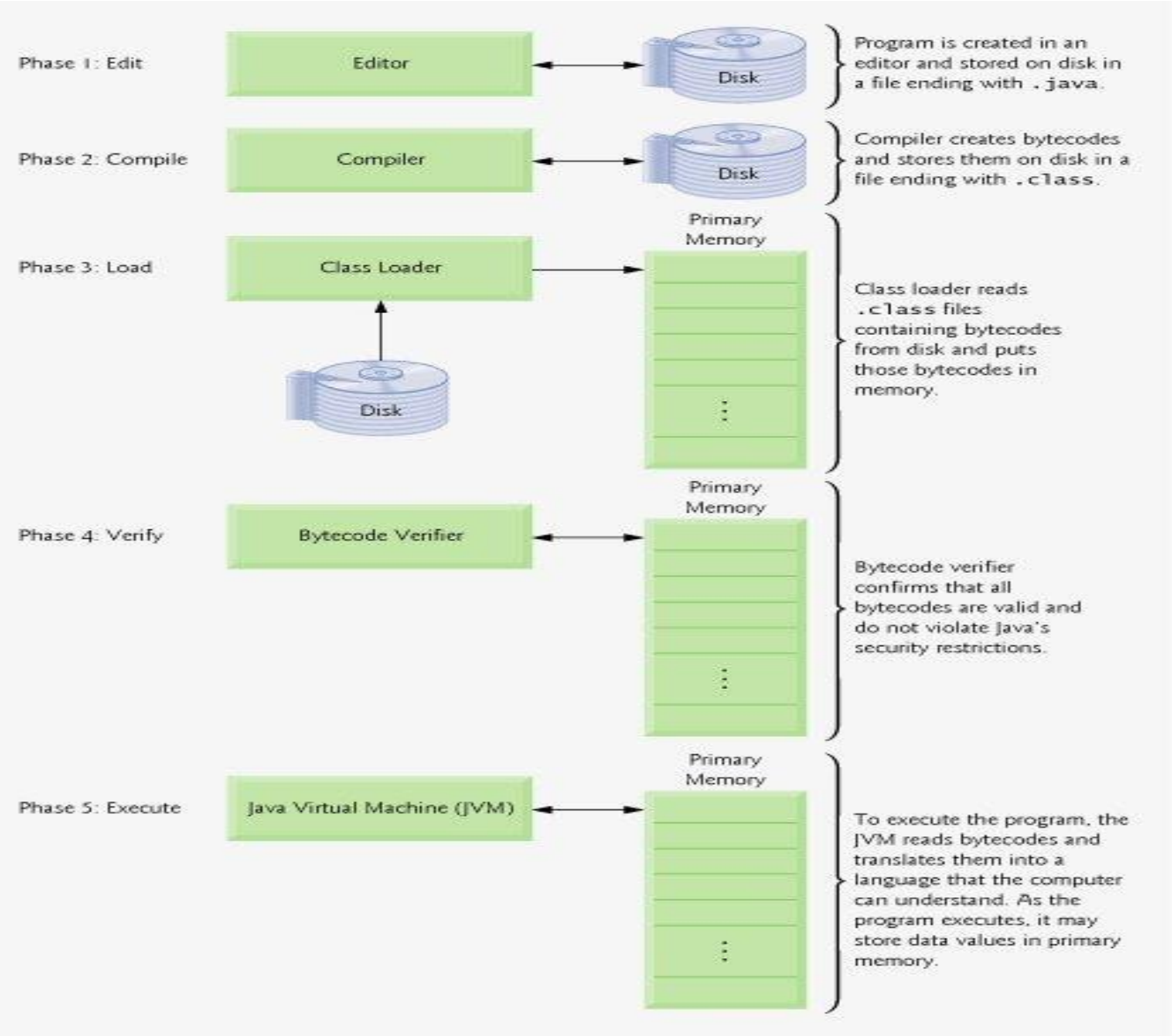
Java Platform Independent Compiling



Five Phases in Java Programs

- Java programs normally go through five phases :
  1. Edit
  2. Compile
  3. Load
  4. Verify
  5. Execute.

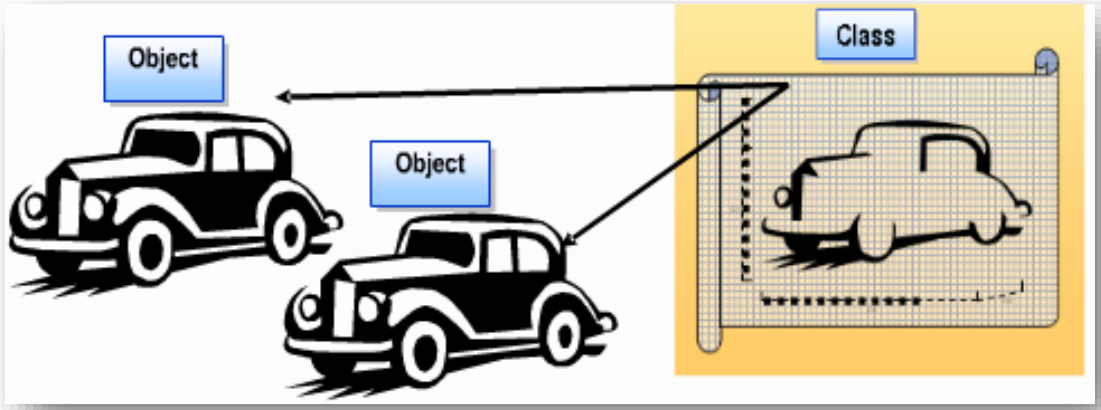
Five Phases in Java Programs



Classes & Objects: Defining, Creating, and Using

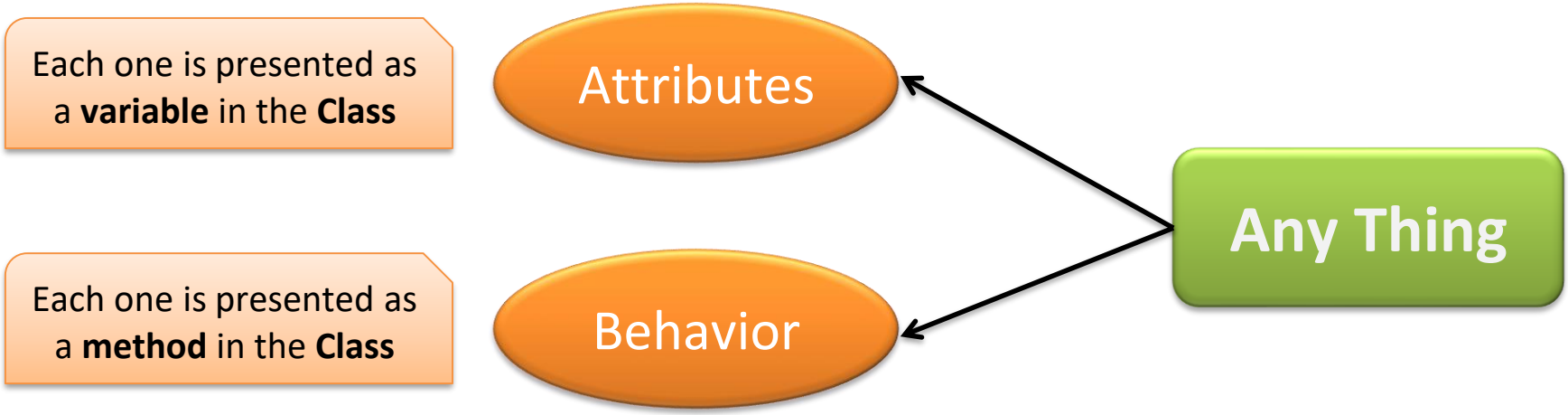
Classes

- A class is a description of a kind of object (known as blue print).
- When we write a program in Java, we define classes, which in turn are used to create objects of the same type.

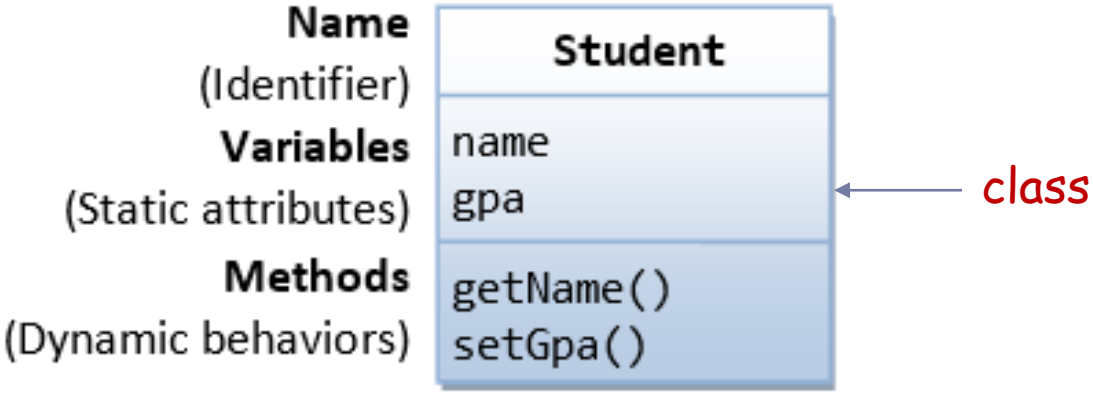


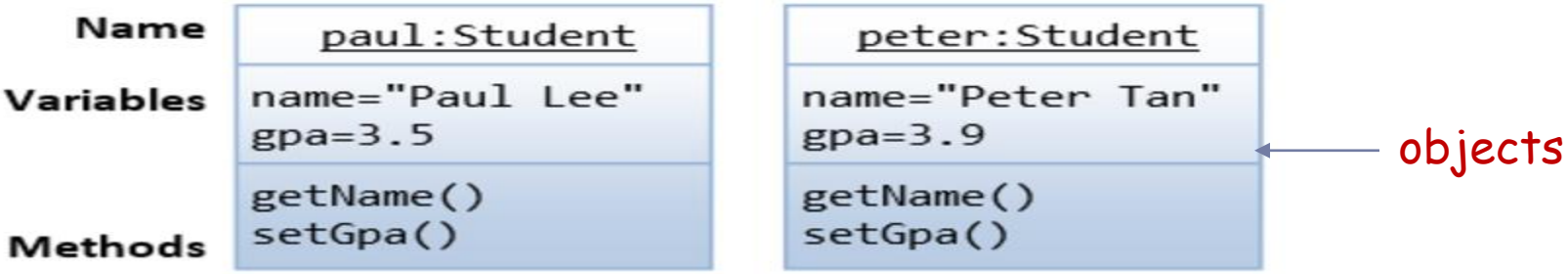
- Classes are, therefore, the main building blocks of Java programs, as everything (objects, attributes and operations) is contained in classes.

Classes


Each one is presented as a **variable** in the **Class**
Attributes
Each one is presented as a **method** in the **Class**
Behavior
Any Thing

Classes


Name
(Identifier)
Variables
(Static attributes)
Methods
(Dynamic behaviors)
Student
name
gpa
getName()
setGpa()
class


Name
Variables
Methods
paul:Student
name="Paul Lee"
gpa=3.5
getName()
setGpa()
peter:Student
name="Peter Tan"
gpa=3.9
getName()
setGpa()
objects

Class Declaration

- To declare a new class, we use the keyword ***class***, followed by a non-reserved identifier that names it. A pair of matching open and close brace characters { and } follow and delimit the class's body.

```
class name {
        declarations

        constructor definition(s)

        method definitions
}
```

← attributes and symbolic constants

← how to create and initialize objects

← how to manipulate the state of objects

These parts of a class can actually be in any order

- A class name shall be a noun or a noun phrase made up of several words. All the words shall be initial-capitalized. For example, SoccerPlayer.

Example

```java
class Person {

    String name;
    String sex;
    String job;
    int age;

    void printInfo() {
        System.out.println("Name: " +name);
        System.out.println("Sex: " +sex);
        System.out.println("Job: " +job);
        System.out.println("Age: " +age);
    }

}
```

Class Declaration

- Every .java file has one or more classes. Only one of the classes can be a public class. Keyword public means that the class is available to everybody.
- That public class must have the same name as the .java file, and contains the method ***main*** in it.
- Execution **always** begins with method ***main*** java applications.

```
// Fig. 4.6: GradeBook.java
// GradeBook class that solves class-average problem using
// counter-controlled repetition.
import java.util.Scanner; // program uses class Scanner

public class GradeBook
}
```

Class Declaration

- To better organize your programs, use a separate .java file for the public class containing method ***main***.

- In general, there should be *one* class *per* file. If you organise things that way, then when you search for a class, you know you only need to search for the file with that name.

Objects

- An object is an instance of a class.
- You can create any number of objects from one class.
- To create a object of a class, you have to:
  1. Declare an object identifier (object name/reference) of a particular class.
  2. Construct the object using the "new" operator.

1. Declaring Object References

- To declare an object, we need an object reference variable.

ClassName objectReferenceName;

- The above statement creates a variable "objectReferenceName" which can reference a ClassName object. It does NOT create an object.

- Variable names begin with a lowercase letter, and subsequent words in the name begin with a capital letter.

2. Instantiating Objects

- To create an object, we use the new keyword along with a constructor for the class of the object we wish to create.

- To refer to the object, we "point" an object reference variable to the new object.

objectReferenceName = new ClassName();

- The declaration and instantiation can be combined as follows:

ClassName objectReferenceName = new ClassName();

Example 1:

1. Declaring object references:

```
Laptop  dell ;
```

2. Instantiating objects

```
dell = new Laptop ();
```

constructor

Or

```
Laptop dell = new Laptop();
```

Example 2:

1. Declaring object references:

```
Student  s1 ;
```

2. Instantiating objects

```
s1 = new Student ();
```

constructor

Or

```
Student s1 = new Student ();
```

Example 2:

StudentRecord.java × Student.java ×

```java
package studentrecord;

public class StudentRecord {

public static void main(String[]args){

        Student s1 = new Student();

}
    }
```

Output - Student (run)

```
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Accessing Members of a Class

- The variables and methods belonging to a class are formally called member variables and member methods.

- To access a member variable or method of a class, you must: First identify the object you are interested in, and then, Use the dot operator (.) to access the desired member variable or method.

Referencing variables:

objectReferenceName.varName

Calling methods:

objectReferenceName.methodName()

Example

Start Page × StudentRecords.java × Student.java ×

```java
package studentrecords;

public class Student {
String name;
String sex;
int age;

void printInfo () {

System.out.println ("Name:" +name);
System.out.println ("Sex:" +sex);
System.out.println ("Age:" +age);
System.out.println ();

}
}
```

Example

Start Page | StudentRecords.java | Student.java

```java
package studentrecords;

public class StudentRecords {

    public static void main(String[] args) {
    Student s1 = new Student ();
    Student s2 = new Student ();

    s1.name = "Nourah";
    s1.sex = "Female";
    s1.age = 19;

    s2.name = "Ahmed";
    s2.sex = "Male";
    s2.age = 21;

    s1.printInfo ();
    s2.printInfo ();
    }

}
```

Output - StudentRecords (run)

```
run:
Name:Nourah
Sex:Female
Age:19

Name:Ahmed
Sex:Male
Age:21

BUILD SUCCESSFUL (total time: 0 seconds)
```

Assignment One is Uploaded to the Portal of “Learning”