

# Introduction to Java Programming

"Overview and basic programming constructs"

## Advanced Programming

Shakirullah Waseeb  
shakir.waseeb@gmail.com

Nangarhar University

February 21, 2017



# Agenda

- 1 Overview
  - Java Language
  - Object Technology Review
- 2 A Typical Java Development Environment
- 3 Basic Java Programming Constructs
  - Simple Java Program
  - Data Types, Variables, and Array
- 4 Questions and Discussion



# Why so many languages?

Language **evolution**, **innovation** and **development** occurs for two fundamental reasons:

- To adapt to changing environments and uses
- To implement refinements and improvements in the art of programming



# Agenda

- 1 Overview
  - Java Language
  - Object Technology Review
- 2 A Typical Java Development Environment
- 3 Basic Java Programming Constructs
  - Simple Java Program
  - Data Types, Variables, and Array
- 4 Questions and Discussion



# Java introduction

- Java is conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991.
- widely used programming language (handheld devices, network, computers)
- Java editions: Standard Edition (SE), Enterprise Edition (EE), Micro Edition (ME)



# Java Buzzwords

- Java is:
  - Simple (inherit C and C++ syntax, adopted by C#)
  - Secure (confining java program to java execution environment )
  - Robust (auto memory management, error handling)
  - Portable (platform independent, byte code, JVM)
  - Object oriented (pure object oriented paradigm)
  - Multithreaded (do many things simultaneously)
  - Distributed (client/server programming, RMI)



# Agenda

- 1 Overview
  - Java Language
  - Object Technology Review
- 2 A Typical Java Development Environment
- 3 Basic Java Programming Constructs
  - Simple Java Program
  - Data Types, Variables, and Array
- 4 Questions and Discussion



# Review on Object Technology

- Demands for new and powerful software: where **quickness**, **economy**, and **correctness** remains an **elusive goal**
- Objects are instances (single occurrence) of **classes** which are essentially *reusable software components*

## Examples

Date object, time object, audio object, video object, people object etc.

- Almost any **noun** can be reasonably represented as a software object in terms of **attributes** (e.g., name, color and size) and **behaviors** (e.g., calculating, moving and communicating).





# Review on Object Technology – Continue

## Bank account example

### Account Class

Attributes : `account_balance`, `date_opened`, `account_type`

Functions : *`inquireBalance`*, *`depositAmount`*, *`withdrawAmount`*

- Instantiation: The process of creating objects of a class, object being created is referred to as instance of that class.
- Reusability: Reuse of existing classes when building new classes and programs save **time and efforts**, also helps in building reliable and effective systems; because they **passed** extensive **testing, debugging and performance** tuning



# Review on Object Technology – Continue

- Messages and Methods calls: Sending message to an object; message is implemented as *method call*
- Encapsulation: wrapping of attributes and methods into objects
- Inheritance: creating *new classes quickly and conveniently* by *absorbing* the characteristics of an existing class, possibly *customizing* them and *adding* unique characteristics of its own



# Review on Object Technology – Continue

- Creating best solution requires:
  - detailed *analysis* in order to
  - determine project's *requirements* (i.e defining *what* the system is supposed to do)
  - and developing a *design* (i.e deciding *how* the system should do it) that satisfies them
- Object-Oriented Analysis and Design (OOAD): analyzing and designing system form object-oriented point of view
- Single graphical language is used for communicating the *results* of any OOAD process, known as *Unified Modeling Language* (UML)
- UML: graphical language for *modeling* object-oriented systems



# Creating and Executing Java application

- Java program creation and execution normally go through five phases – edit, compile, load, verify, and execute
- We discuss these phases in the context of the Java SE Development Kit (JDK) <sup>1</sup>

---

<sup>1</sup>[www.oracle.com/technetwork/java/javase/downloads/index.html](http://www.oracle.com/technetwork/java/javase/downloads/index.html)



# Creating and Executing Java application - Phase 1

- Use an editor (vi, emacs on linux, notepad in windows)
- Type your java program typically referred as source code
- Save file with **.java extension**



[1]



## Creating and Executing Java application - Phase 2

- Use command **javac** (java compiler) to compile the source program
- For example a program called **Sallam.java** we use following command **javac Sallam.java**
- If program compiles successfully it will produce **Sallam.class** file which is the compiled version of program and called **bytecode**

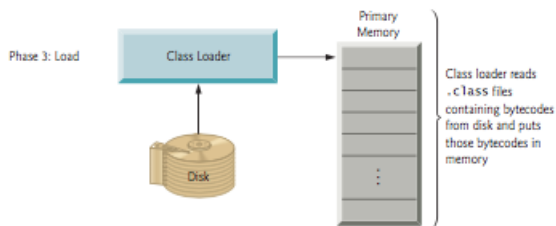


[1]



## Loading a Program into Memory - Phase 3

- **JVM** places program in memory to execute it – known as **loading**
- **JVM's class loader** takes the .class files into memory (also .class files that the program uses)
- These .class files can be loaded from hard disk or from network

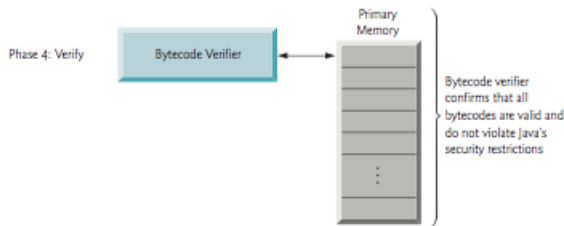


[1]



# Bytecode Verification - Phase 4

- **Bytecode Verifier** examines their bytecodes to **ensure** that they're **valid** and **do not violate Java's security restrictions**
- **Java enforces strong security**



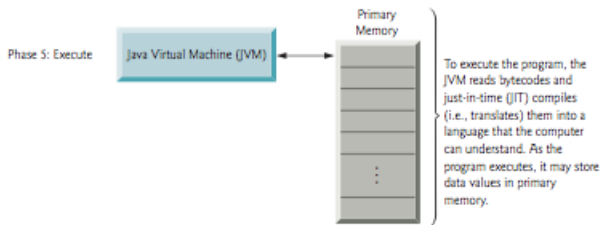
[1]





## Execution - Phase 5

- **JVM** executes the bytecode
- Performing specified actions in the program
- These .class files can be loaded from hard disk or from network



[1]



# Agenda

- 1 Overview
  - Java Language
  - Object Technology Review
- 2 A Typical Java Development Environment
- 3 Basic Java Programming Constructs
  - Simple Java Program
  - Data Types, Variables, and Array
- 4 Questions and Discussion



# Simple Java Program

## A Simple Program Example

```
class MyClass {  
    public static void main (String args[]){  
        System.out.println("Assalam-o-Alikum");  
    }  
}
```

- Save it as **MyClass.java**
- Compile it as: **javac MyClass.java**
- Execute it as: **java MyClass**



# Agenda

- 1 Overview
  - Java Language
  - Object Technology Review
- 2 A Typical Java Development Environment
- 3 Basic Java Programming Constructs
  - Simple Java Program
  - Data Types, Variables, and Array
- 4 Questions and Discussion



# Data Types

- Java is strongly typed language

Note: every variable has a type, every expression has a type, and every type is strictly defined

All assignments, whether explicit or via parameters passing in methods call are checked for type compatibility

- There are no automatic coercions or conversions of conflicting types as in some languages.
- Simple Types
  - Integers: byte, short, int, long
  - Floating point: float, double
  - Characters: char
  - Boolean: boolean



# Variables

- Variables are defined via a combination of **type**, **identifier**, and an **optional initializer**  
**type identifier** [ **= value** ] [, **identifier** [**= value**] ...] ;
- Dynamic initialization
- Scope and life time of variables; two general categories **global** and **local**, however java define **class** and **method** scope
- Type conversion and type casting
  - Automatic conversion: takes place if; **two types are compatible**, **destination type is larger than source type**
  - Casting incompatible type: a cast is used to make a conversion between incompatible types: **narrows conversion** (e.g casting a large value type into small value type **int to byte**) **truncation** (e.g converting float type into integer type)  
**(target-type) value**
- Automatic type promotion and promotion rules



# Arrays

- An **array** is a group of **like-typed variables** that are referred to by a **common name**.
- One dimensional array: The general form of a one dimensional array declaration is

*type array-var[ ];* no memory will set aside

*array-var = new type[size];* memory of given size will be reserved

*array-var[0] = value1; array-var[1] = value2;*

*array-var[ ] = {value1, value2};*

- Two dimensional array: The general form of declaration is

*type array-var[ ][ ];*

*array-var = new type[row-size][column-size];*

*array-var[0][0] = value1; array-var[0][1] = value2; array-var[1][0] =*

*value3; array-var[1][1] = value4;*

*array-var[ ][ ] = { {value1, value2}, {value3, value4} };*



# Your Turn: Time to hear from you!



2



<sup>2</sup><https://fensafitters.files.wordpress.com/2013/07/3d095.jpg>



# References

-  P.J. Deitel, H.M. Deitel  
*Java How to program, 9th Edition* .  
Prentice Hall, 2012.
-  Herbert Schildt  
*The complete reference Java2, 5th Edition* .  
McGraw-Hill/Osborne, 2002.

