Computer Programming
Essentials| Java | Java Technologies

Taras Matyashovskyy

Agenda

- Computer Programming
- Compilation vs. Interpretation
- History of Java
- Main Features of Java
- JDK
- History of Releases
- Java Platforms
- Java Technologies
- Examples of Java Projects

Computer Programming

Compilation vs. Interpretation

Computer Programming

Programming – process that leads from an original formulation of a computing problem to executable programs.

Usually it includes:
- **analysis**, **understanding**, and generically **solving** such problems resulting in an algorithm

- **coding** of the algorithm in a target programming language

- **testing**, **debugging**, and **maintaining** the source code

Computer Programming

Creating a sequence of instructions
to enable the computer to do something

Computer Programming Algorithm

The algorithm is often only represented
in human-parseable form and reasoned about using logic.

Source code is written in one or more programming
languages, e.g. C++, C#, Java, Python, JavaScript, etc.

Compilation

Translation of source code into machine code

Programmer writes the code

Compiler transforms it to machine code
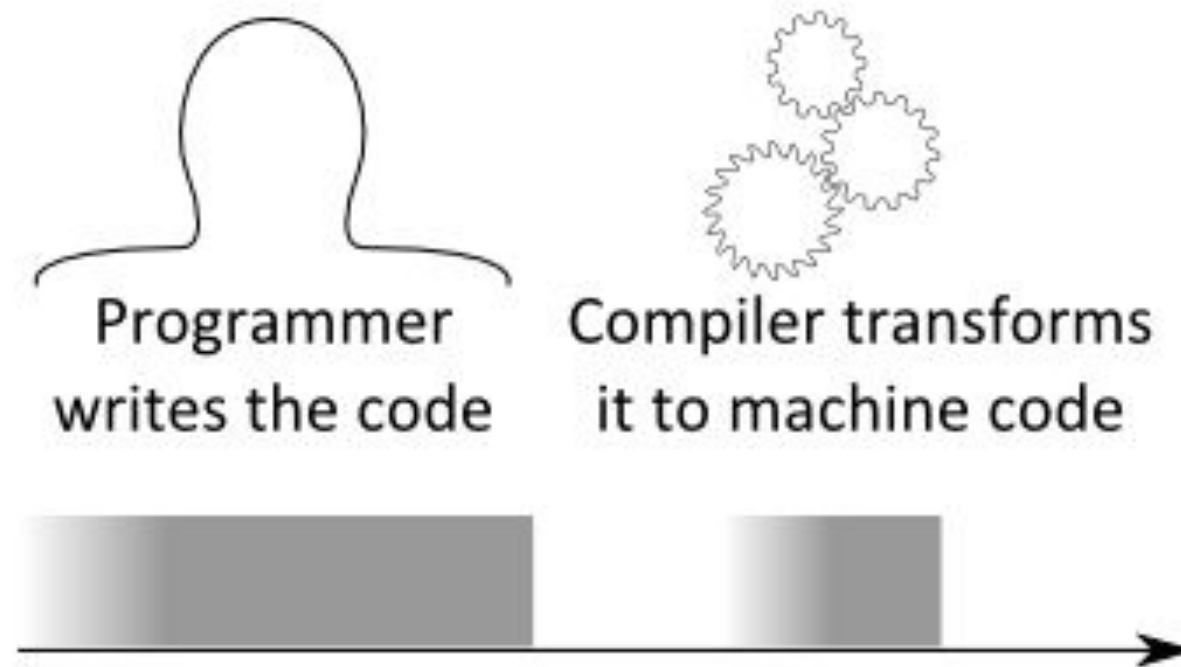
Compiled Language

A compiled language is one where the program, once compiled, is expressed in the instructions of the **target machine**.

For example, an addition "+" operation in your source code could be translated directly to the "ADD" instruction in machine code.
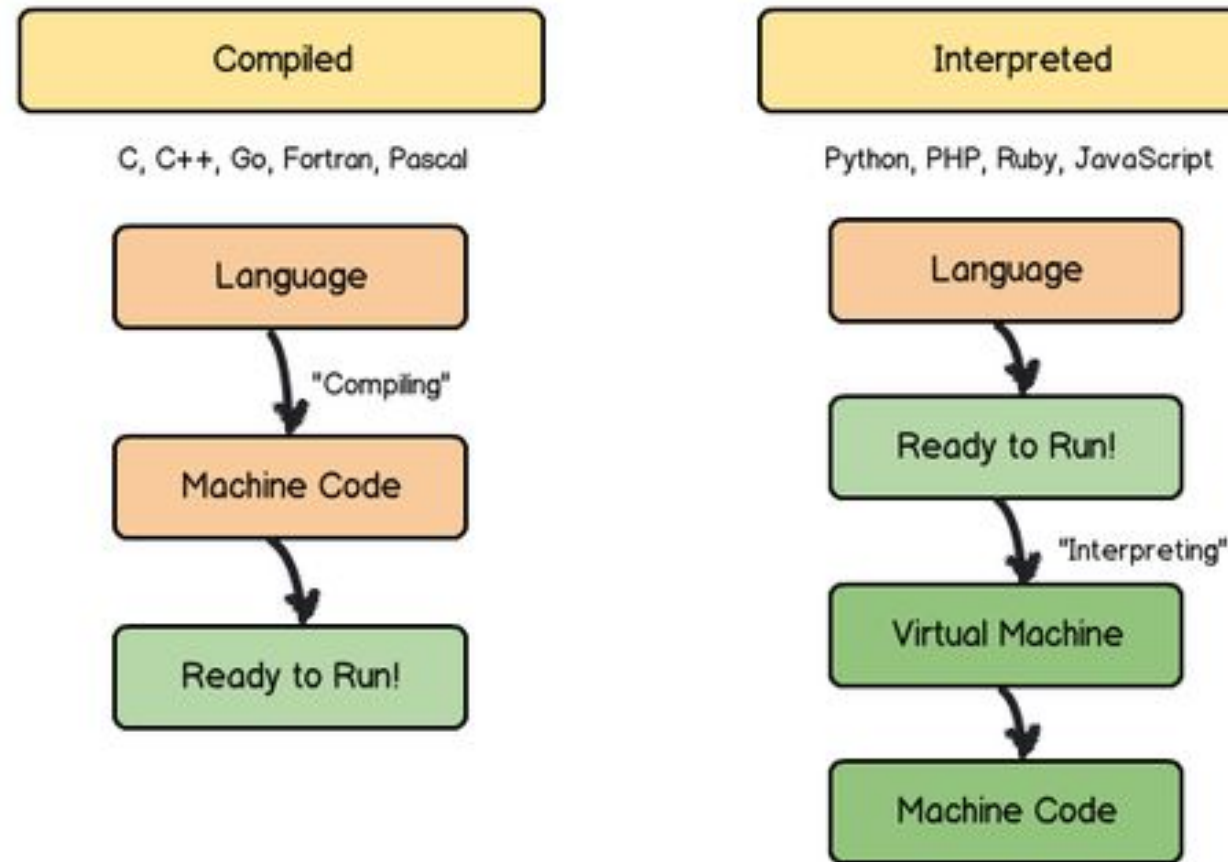
Interpreted Language

An interpreted language is one where the instructions are not directly executed by the the target machine, but instead read and executed by some other program (which normally is written in the language of the native machine).

For example, the same "+" operation would be recognized by the interpreter at run time, which would then call its own "add(a,b)" function with the appropriate arguments, which would then execute the machine code "ADD" instruction.

Compiler vs. Interpreted

Compiler Language

Pros:

- Faster performance by directly using the native code of the target machine
- Opportunity to apply quite powerful optimizations during the compile stage

Cons:

- writing good compilers is very hard

Interpreted Language

Pros:

- Easier to implement
- No need to run a compilation stage, i.e. can execute code directly "on the fly"
- Can be more convenient for dynamic languages

Cons:

- Slow to execute because each statement had to be interpreted into machine code every time it was executed

Why is This Important to Recruiter?

- Programming language is just a tool, but usually Ukrainian developers are tight to specific language, sometimes first language, and stick with it for a long time

- Identity and preferences to specific programming language can sometimes explain professional philosophy of the engineer

Java

History of Java. Main Features of Java. JDK. History of Releases.

History of Java

Java language was originally developed by James Gosling at Sun Microsystems, which is now a subsidiary of Oracle Corporation, and released in 1995 as a core component of Sun Microsystems' Java platform.

Java has derived much of its syntax from C/C++, but it has fewer low-level facilities than either of them.

Main Features

There were 5 primary goals in the creation of the Java language.

It should:
- be **simple**, **object-oriented** and **familiar**
- **robust** and **secure**
- be **architecture-neutral** and **portable**
- execute with **high performance**
- be **interpreted**, **threaded**, and **dynamic**

Simple, Object-oriented and Familiar

Java can be programmed without extensive programmer training.

The needs of distributed, client-server based systems coincide with the encapsulated, message-passing paradigms of object-based software.

It looks like C++ as far as possible results in it being a familiar language, while removing the unnecessary complexities of C++.

Robust & Secure

Java provides extensive compile-time checking, followed by a second level of run-time checking.

Java technology is designed to operate in distributed environments, which means that security is of paramount importance.

Architecture Neutral and Portable

Java is intended to let application developers **"write once, run anywhere"**, meaning that code that runs on one platform does not need to be recompiled to run on another

Architecture Neutral and Portable

Java applications are compiled to **byte code** (.class file) –architecture neutral intermediate format designed to transport code efficiently to multiple hardware and software platforms.

Byte code can run on any **Java virtual machine** (**JVM**) regardless of computer architecture.

JVM is a virtual machine, i.e. a machine running inside your real machine (Windows, Linux, Mac) written specifically for the host hardware.

Architecture Neutral and Portable
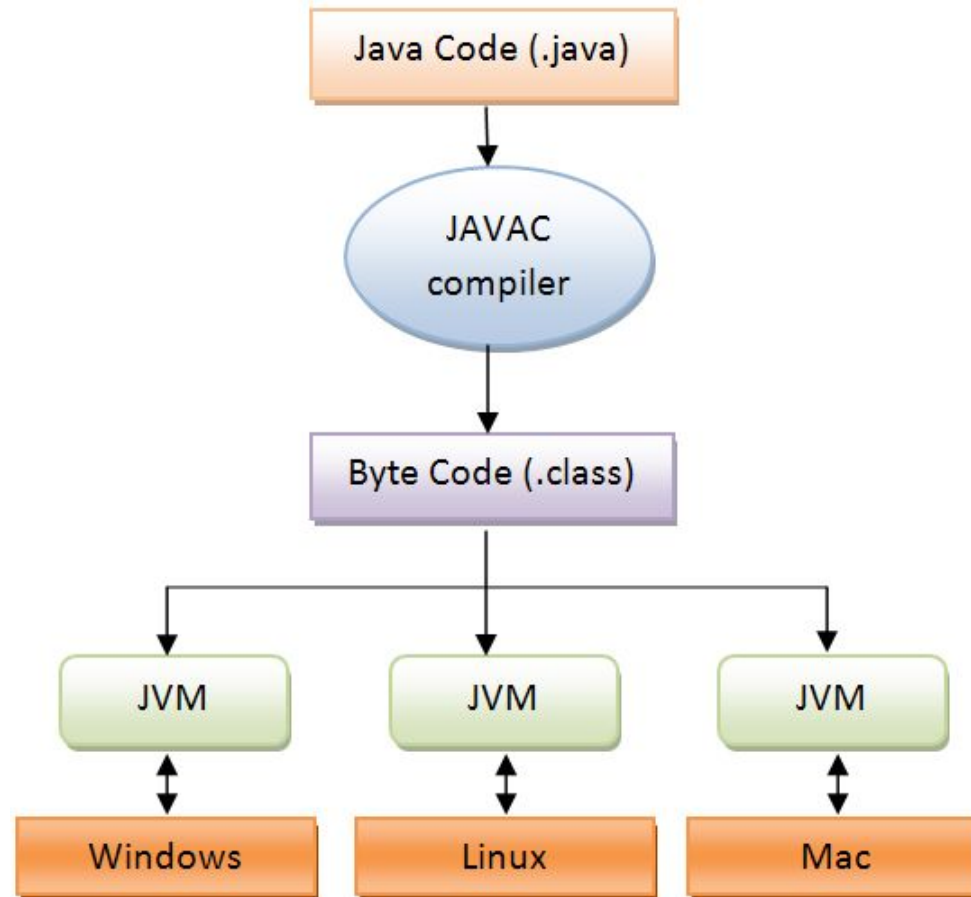
High Performance

Java has a lot of optimization techniques:
- Just-In-Time compilation (the program is stored in memory as **byte code**, but the code segment currently running is preparatively **compiled** to physical machine code in order to run faster)
- Adaptive optimization
- Garbage Collection

High Performance

Java performance is generally:
- **slower** than compiled languages such as C or C++
- **similar** to other Just-in-time compiled languages such as C#
- **much faster** than languages without an effective native-code compiler (JIT or AOT), such as Perl, Ruby, PHP and Python

Java Development Kit

JDK contains tools for developing, debugging, and monitoring Java applications. For instance:
- *javac* – the Java compiler, which converts source code into Java byte code
- *java* – the loader for Java applications, i.e. java interpreter
- *jar* – the archiver, which packages related class libraries into a single JAR file

Java Version History

JDK Alpha and Beta (1995)

JDK 1.0 (January 23, 1996)

JDK 1.1 (February 19, 1997)

Java 2 splits into J2SE, J2EE, J2ME

J2SE 1.2 (December 8, 1998)

J2SE 1.3 (May 8, 2000)

J2SE 1.4 (February 6, 2002)

J2SE 5.0 (September 30, 2004)

Java SE 6 (December 11, 2006)

Java SE 7 (July 28, 2011)

Java SE 8 (March 18, 2014)

Java SE 8 Update 66 (October 21, 2015)

Java Platform

Consists of distinct, but interrelated technologies:
- The Java Virtual Machine (JVM)
- Class loaders and class files
- The JVM languages, e.g. Java, Groovy, Scala, Closure
- The APIs

The set of APIs is controlled by Sun Microsystems in cooperation with others through the Java Community Process program.

Java Platform Editions

- **Java Card –** a technology that allows small Java-based applications to be run securely on smart cards and similar small-memory devices

- **Java ME (Micro Edition) –** specifies several different sets of libraries for devices with limited storage, display, and power capacities. Often used to develop applications for mobile devices, PDAs, TV set-top boxes, and printers.

Java Platform Editions

- **Java SE (Standard Edition) –** for general-purpose use on desktop PCs, servers and similar devices

- **Java EE (Enterprise Edition) –** test Java SE plus various APIs useful for multi-tier client–server enterprise applications

Why is This Important to Recruiter?

Java is just an example, but good recruiter needs to:

- know specificity of the language he/she is hiring for
- know latest version of the language, so he/she should understands relevance and freshness of the project for which he/she is hiring
- use correct naming in the job description
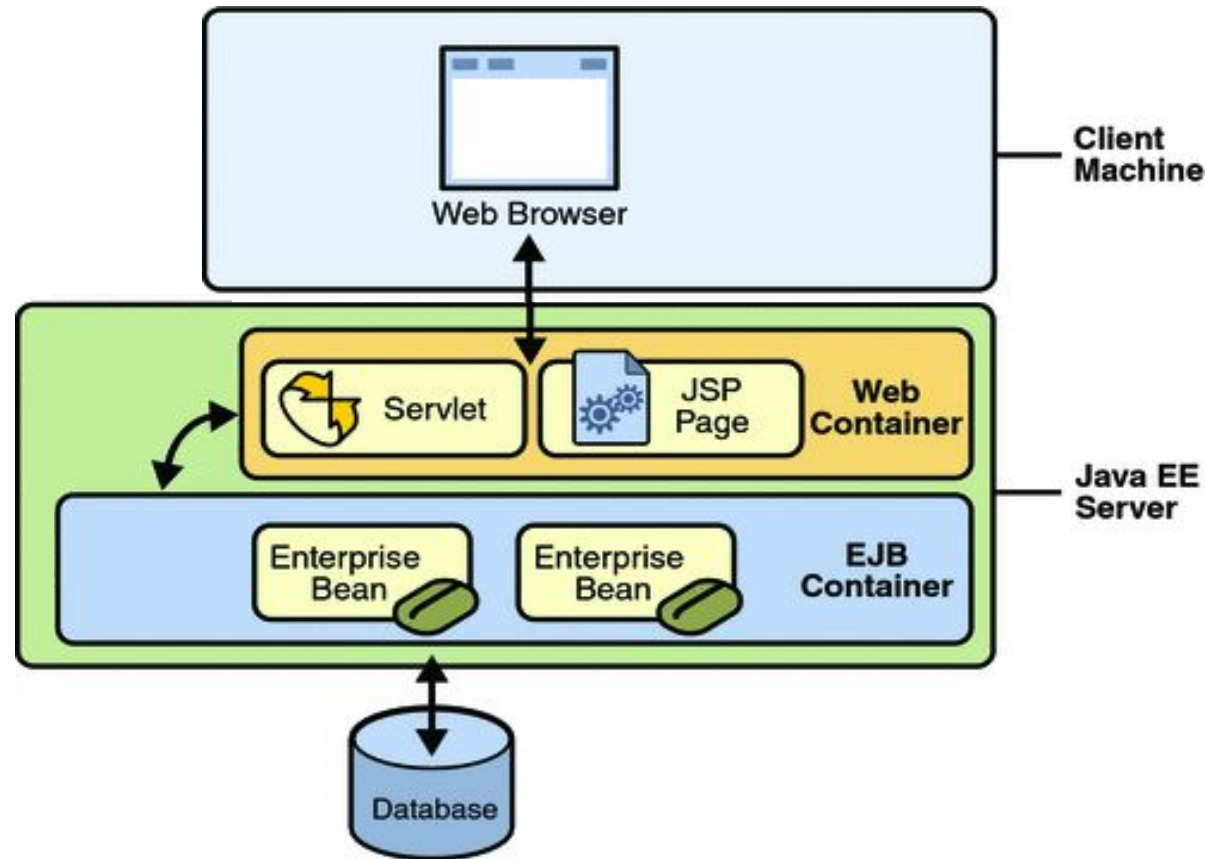- etc.

Java Technologies

Servlet. JSP. JDBC. Hibernate. Swing. Spring Framework.

Web Application

Java Web Application Request Handling
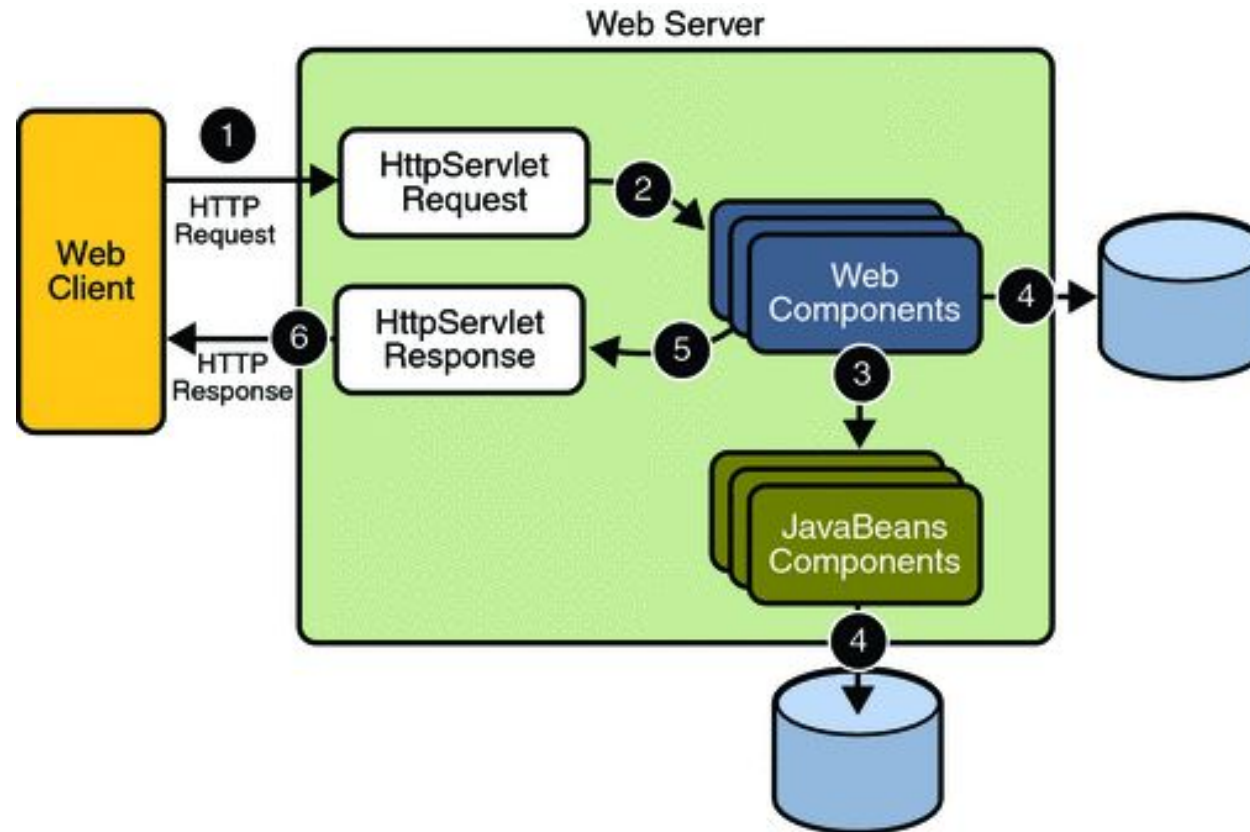
Web Application

Servlet

- Java programming language class used to extend the capabilities of a server by providing dynamic web content

- part of Java SE

- to deploy and run servlet, a compatible web server with a **servlet container**, such as Apache Tomcat or Jetty, is required

Java Server Pages

- technology that helps to create **dynamically** generated web pages based on HTML, XML

- JSP allows dynamic Java code to be combined with static web markup content, with the resulting page being compiled and executed on the server to deliver a **dynamic document**

- to deploy and run JavaServer Pages, a compatible web server with a **servlet container**, such as Apache Tomcat or Jetty, is required

Java Database Connectivity (JDBC)

- is an API for the Java programming language that defines how a client may access a database

- provides methods for querying and updating data in a database

- is a part of Java SE

- is oriented towards relational databases

- supports multiple drivers, e.g. for MySQL, Oracle, Postgres, etc.

Hibernate

- object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database.

- generates SQL calls and relieves the developer from manual result set handling and object conversion

Swing

Primary Java GUI widget toolkit
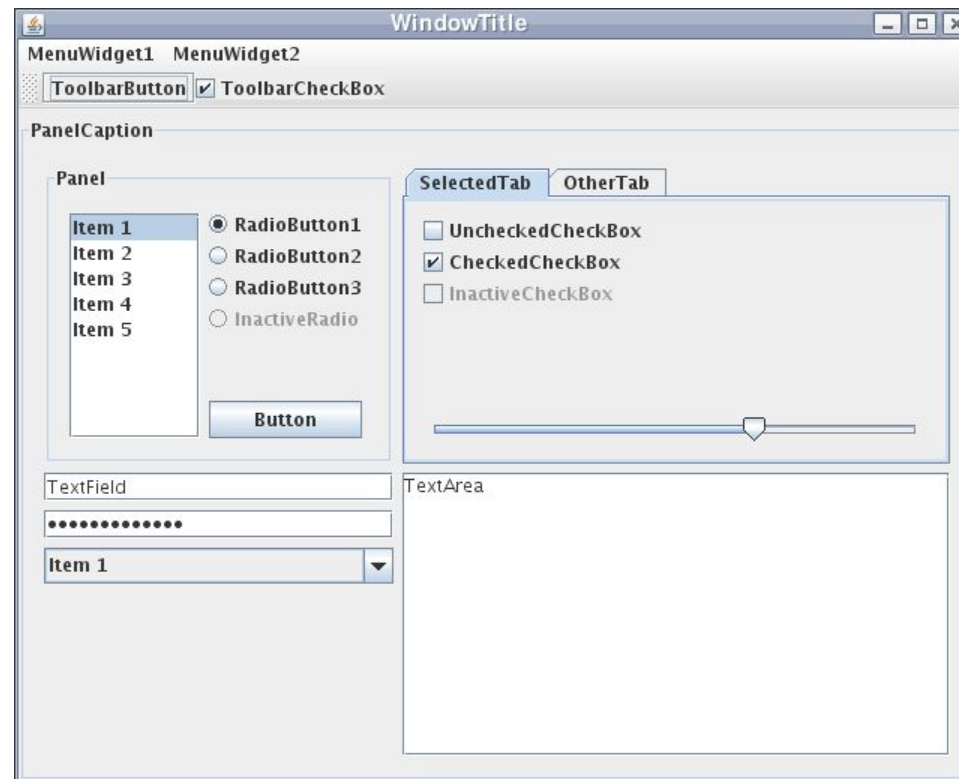
Spring Framework

- is an open-source application framework and inversion of control container for the Java platform

- has become popular in the Java community as an alternative to the Enterprise JavaBean (EJB) model used in JEE Platform

- has a lot of modules

Why is This Important to Recruiter?

Java technology stack is just an example, but good recruiter needs to:

- read and understand candidate's technical profile

- understand high-level project requirements

- read, understand and/or write correct job description

- etc.

Examples of Java Projects

Real anonymized descriptions

Project #1

- Web application with user interface
- Main technologies:
  - Java 6.x, Javascript
  - JDBC, Hibernate
  - Spring 3.x
  - Spring MVC, JSP
  - Spring RESTful services
  - HTML/CSS/JS, JQuery, Raphael.js, AngularJS
- Database: MySQL
- Servers: Jetty
- Build Tools: Maven
- Version Control: Git

Project #2

- Web application
- Main technologies:
  - Java 6.x, C++, CORBA
  - Hibernate, JDBC
  - EJB 3
  - Spring
  - Swing
- Database: Oracle, H2
- Servers: JBoss, Tomcat
- Build Tools: Ant, Maven
- Version Control: SVN

Use real world and practice!

Let us find something interesting

on DOU.ua, rabota.ua, hh.ua, etc?!