Java: An overview

CSC02A2

Outline

Outline

1. Programming Languages
   - Programming language categories
   - Conventional compilation vs. JVM
2. Java Technologies
   - Java over the years
   - Design Goals of the Java Language
   - Java Features
   - Java Editions
3. Java and C++ compared
   - Pre-processor
   - Data Structures
   - Domains and Visibility
   - Pointers
   - Primative Data Types
   - Garbage Collection
   - Exception Handling
   - Inheritance
   - Operator Overloading
   - Interface
   - Object Allocation
   - Multithreading
   - Strings and Arrays

Programming Languages

Programming language categories I

Low-level language - Machine code

- Hardware specific.
- Purely numerical (binary numbers represented both data and instructions)

```
10111010111010100100101100100101
01010111010010101101010111010101
10100101011110101101010001011101
11101110110100110011101010100100
```

Low-level language - Assembly Language

- Hardware specific.
- Machine code instructions are represented by alphanumeric identifiers.
- Easier to use than pure machine code.
- Needs to be assembled (compiled) into machine code prior to execution.

Outline
Programming Languages
Programming language categories
Conventional compilation vs. JVM
Java Technologies
Java over the years
Design Goals of the Java Language
Java Features
Java Editions
Java and C++ compared
Pre-processor
Data Structures
Domains and Visibility
Pointers
Primative Data Types
Garbage Collection
Exception Handling
Inheritance
Operator Overloading
Interface
Object Allocation
Multithreading
Strings and Arrays

Programming language categories II

High-level languages - General

- Allows high level programming paradigms (structured programming, object orientation etc.)

High-level languages - Compiled

- Possible to write portable code (back end of compiler can often target multiple platforms)
- Textual source code is compiled into object code and then linked to form executables.

High-level languages - Interpreted

- Source code is parsed (read-in) and then directly executed by a software system referred to as an interpreter.
- Heavy performance penalty
- Forces distribution of source code.

Outline

Programming Languages

Programming language categories

Conventional compilation vs. JVM

Java Technologies

Java over the years

Design Goals of the Java Language

Java Features

Java Editions

Java and C++ compared

Pre-processor

Data Structures

Domains and Visibility

Pointers

Primative Data Types

Garbage Collection

Exception Handling

Inheritance

Operator Overloading

Interface

Object Allocation

Multithreading

Strings and Arrays

Programming language categories III

High-level languages - Byte code interpreted

- Source form is translated into a machine independent compiled form (byte code or p-code) which is then executed by a software implemented virtual machine.
- Virtual machine provides a single target platform, thereby promoting portability.
- Virtual machine adds a performance overhead as it interprets the byte code.
- Source code need not be distributed, although reverse engineering is possible.

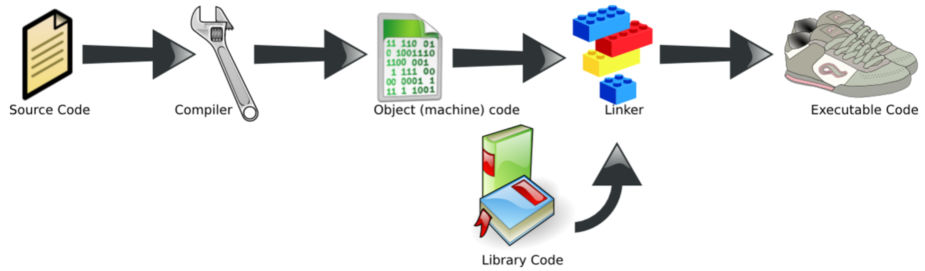
Outline

Programming Languages
Programming language categories
Conventional compilation vs. JVM

Java Technologies
Java over the years
Design Goals of the Java Language
Java Features
Java Editions

Java and C++ compared
Pre-processor
Data Structures
Domains and Visibility
Pointers
Primative Data Types
Garbage Collection
Exception Handling
Inheritance
Operator Overloading
Interface
Object Allocation
Multithreading
Strings and Arrays

Conventional Compilation



Outline

Programming Languages

Programming language categories

Conventional compilation vs. JVM

Java Technologies

Java over the years

Design Goals of the Java Language

Java Features

Java Editions

Java and C++ compared

Pre-processor

Data Structures

Domains and Visibility

Pointers

Primative Data Types

Garbage Collection

Exception Handling

Inheritance

Operator Overloading

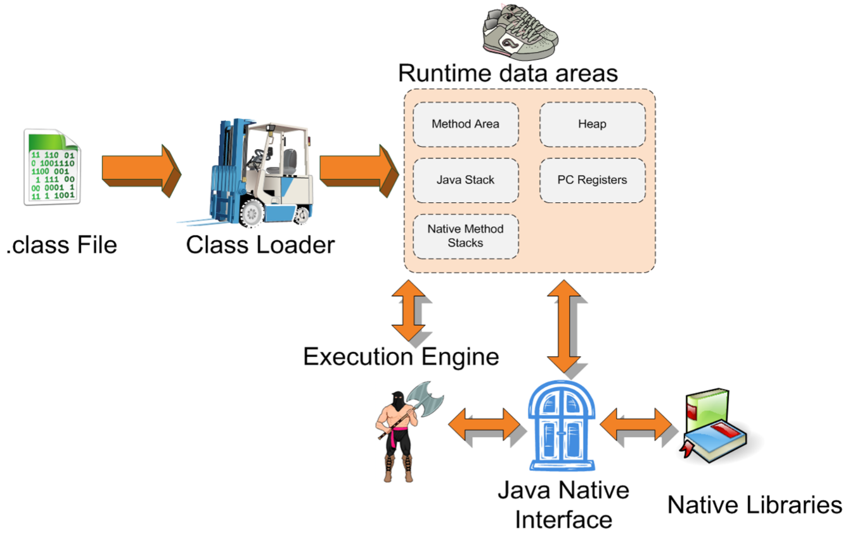
Interface

Object Allocation

Multithreading

Strings and Arrays

JVM Architecture



Outline

Programming Languages
Programming language categories
Conventional compilation vs. JVM

Java Technologies
Java over the years
Design Goals of the Java Language
Java Features
Java Editions

Java and C++ compared
Pre-processor
Data Structures
Domains and Visibility
Pointers
Primative Data Types
Garbage Collection
Exception Handling
Inheritance
Operator Overloading
Interface
Object Allocation
Multithreading
Strings and Arrays

Java Technologies

Design Goals of the Java Language

- Simple, object oriented, and familiar.
- Robust and secure.
- Architecture neutral and portable.
- High performance.
- Interpreted, threaded and dynamic.

Outline

Programming Languages

Programming language categories

Conventional compilation vs. JVM

Java Technologies

Java over the years

Design Goals of the Java Language

Java Features

Java Editions

Java and C++ compared

Pre-processor

Data Structures

Domains and Visibility

Pointers

Primative Data Types

Garbage Collection

Exception Handling

Inheritance

Operator Overloading

Interface

Object Allocation

Multithreading

Strings and Arrays

Java Features

Object orientation
: Divides programs into separate objects encapsulating both behaviour and data.

Robust
: Errors in programs rarely cause system crashes. Pre-runtime error detection.

Secure
: Protection against untrusted code via sandboxing.

Distributed
: Designed to run on computer networks. Language and library support.

Platform independence
: Portability across all machines for which the java runtime environment exists.

Outline

Programming Languages
- Programming language categories
- Conventional compilation vs. JVM

Java Technologies
- Java over the years
- Design Goals of the Java Language
- Java Features
- Java Editions

Java and C++ compared
- Pre-processor
- Data Structures
- Domains and Visibility
- Pointers
- Primative Data Types
- Garbage Collection
- Exception Handling
- Inheritance
- Operator Overloading
- Interface
- Object Allocation
- Multithreading
- Strings and Arrays

Java Editions

- Java Standard Edition (SE) – The primary Java release. Suitable for most application development.
- Java Enterprise Edition (EE) – Used in large scale corporate environments. Adds capabilities such as:
    - EJB (Enterprise Java Beans – Java Components)
    - XML streams
    - Persistence (long term object storage)
    - Various packages related to interfacing with the enterprise messaging system.
- Java Micro Edition (ME) – Used to write software for mobile phones, set-top boxes, cards and other embedded devices.

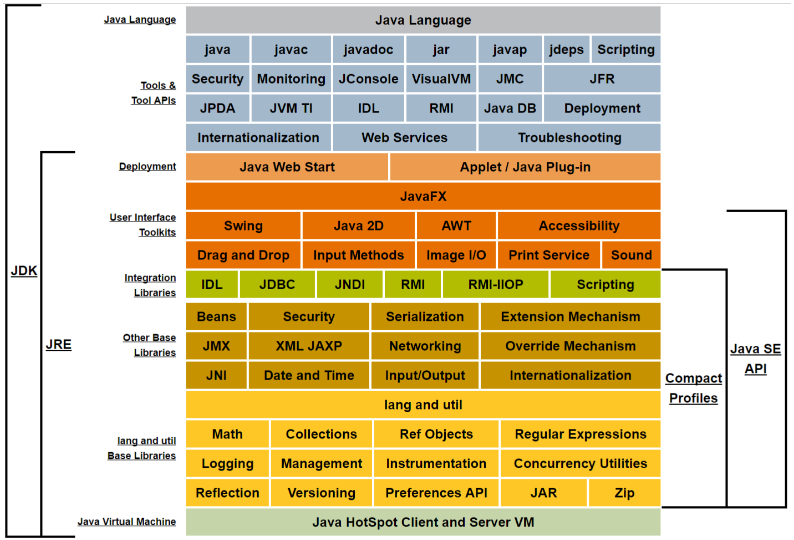
Outline

Programming Languages
Programming language categories
Conventional compilation vs. JVM

Java Technologies
Java over the years
Design Goals of the Java Language
Java Features
Java Editions

Java and C++ compared
Pre-processor
Data Structures
Domains and Visibility
Pointers
Primative Data Types
Garbage Collection
Exception Handling
Inheritance
Operator Overloading
Interface
Object Allocation
Multithreading
Strings and Arrays

Java SE at a glance



Java and C++ compared

Differences between Java and C++ I

Pre-processor

C++	Java
• Include header statements • Define macros	• No pre-processor available

Data Structures

C++	Java
• Structs (records) • Unions • Classes	• Classes • Enumerations • Interfaces

Outline

Programming Languages
- Programming language categories
- Conventional compilation vs. JVM

Java Technologies
- Java over the years
- Design Goals of the Java Language
- Java Features
- Java Editions

Java and C++ compared
- Pre-processor
- Data Structures
- Domains and Visibility
- Pointers
- Primative Data Types
- Garbage Collection
- Exception Handling
- Inheritance
- Operator Overloading
- Interface
- Object Allocation
- Multithreading
- Strings and Arrays

Differences between Java and C++ II

Domains and Visibility

C++	Java
• Code grouped in namespaces	• Code grouped in packages
• Public	• Public
• Private	• Private
• Protected	• Protected
• Friend	• Package

Pointers

C++

- Programmer must handle pointer manipulation
- Objects passed by reference or by value

Java

- Pointer manipulation done by compiler
- All objects act as pointers
- Only passed by reference

Outline

Programming Languages
- Programming language categories
- Conventional compilation vs. JVM

Java Technologies
- Java over the years
- Design Goals of the Java Language
- Java Features
- Java Editions

Java and C++ compared
- Pre-processor
- Data Structures
- Domains and Visibility
- Pointers
- Primative Data Types
- Garbage Collection
- Exception Handling
- Inheritance
- Operator Overloading
- Interface
- Object Allocation
- Multithreading
- Strings and Arrays

Differences between Java and C++ III

Primative Data Types

C++	Java
• Size of data types are machine specific	• Size of primitive data types fixed

Garbage Collection

C++	Java
• Programmer must remember to free memory	• Automatic garbage collection • No ***`delete`*** available

Exception Handling

C++	Java
• No forced exception handling	• Very strict rules on exception handling

Outline

Programming Languages
- Programming language categories
- Conventional compilation vs. JVM

Java Technologies
- Java over the years
- Design Goals of the Java Language
- Java Features
- Java Editions

Java and C++ compared
- Pre-processor
- Data Structures
- Domains and Visibility
- Pointers
- Primative Data Types
- Garbage Collection
- Exception Handling
- Inheritance
- Operator Overloading
- Interface
- Object Allocation
- Multithreading
- Strings and Arrays

Differences between Java and C++ IV

Inheritance

C++

- Class can inherit methods from more than one class.

Java

- No multiple inheritance.
- Class can be derived from at most one base class.
- Interface implementation used to simulate multiple inheritance

Operator Overloading

C++

- Operator overloading supported

Java

- No operator overloading (some limited exceptions).

Outline

Programming Languages

Programming language categories

Conventional compilation vs. JVM

Java Technologies

Java over the years

Design Goals of the Java Language

Java Features

Java Editions

Java and C++ compared

Pre-processor

Data Structures

Domains and Visibility

Pointers

Primative Data Types

Garbage Collection

Exception Handling

Inheritance

Operator Overloading

Interface

Object Allocation

Multithreading

Strings and Arrays

Differences between Java and C++ V

Interface

C++

- Pure abstract base class.

Java

- Equivalent to C++ class which only contains, pure-virtual functions.
- Class that realises interface must provide implementation.

Object Allocation

C++

- Objects may be Stack or Heap allocated.

Java

- Objects only Heap allocated.

Outline

Programming Languages
- Programming language categories
- Conventional compilation vs. JVM

Java Technologies
- Java over the years
- Design Goals of the Java Language
- Java Features
- Java Editions

Java and C++ compared
- Pre-processor
- Data Structures
- Domains and Visibility
- Pointers
- Primative Data Types
- Garbage Collection
- Exception Handling
- Inheritance
- Operator Overloading
- Interface
- Object Allocation
- Multithreading
- Strings and Arrays

Differences between Java and C++ VI

Multithreading

C++	Java
• Thread capabilities were added in later third-party libraries. • No single standard.	• Built in multi-threading • Standard class libraries were designed to be able to handle threads. • ***synchronized*** keyword.

Strings and Arrays

C++	Java
• string, char*. • Arrays implemented using pointers.	• String (immutable) and StringBuffer classes. • Arrays as Objects (some restrictions).

Outline

Programming Languages
- Programming language categories
- Conventional compilation vs. JVM

Java Technologies
- Java over the years
- Design Goals of the Java Language
- Java Features
- Java Editions

Java and C++ compared
- Pre-processor
- Data Structures
- Domains and Visibility
- Pointers
- Primative Data Types
- Garbage Collection
- Exception Handling
- Inheritance
- Operator Overloading
- Interface
- Object Allocation
- Multithreading
- Strings and Arrays