Programming with Java for Beginners

Bineet Sharma

Introduction

Assumptions & Expectations

Introduction Series Part I

- **My Assumptions**
  - Syllabus
- **Your Expectations**
  - Programming concepts, computer languages
  - Learn to compile and run HelloJavaWorld program

Objectives

Introduction Series Part I

- **Computer Components**
  - Hardware and Software
  - Languages: Natural, Formal, Programming
  - Computer Programs
- **Software Development Process**
  - Software Development Life Cycle
  - Software Development Paradigm
  - Object Oriented Programming Methods
- **Introducing Java**
  - Why Java? How Is It Used?
  - First Java Program

Introducing Java

Introduction Series Part I

```
public class FirstJavaHello {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Helo World and Students of Java!");
	}

}
```

Computer Components

Introduction Series Part I

- **Two Primary Components**
  - **Hardware** is something you touch and see
  - **Software** gives functionality to the hardware

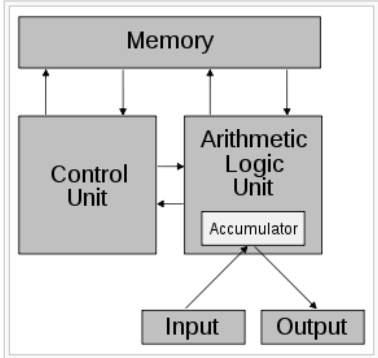
Von Neumann architecture

Introduction Series Part I

- **John von Neumann – Father of modern computers**

In V N Stored-Prog Computer

-Memory holds both instruction and data
-- Interpretation depends on program's logic



Schematic of the von Neumann architecture. The Control Unit and Arithmetic Logic Unit form the main components of the Central Processing Unit (CPU)

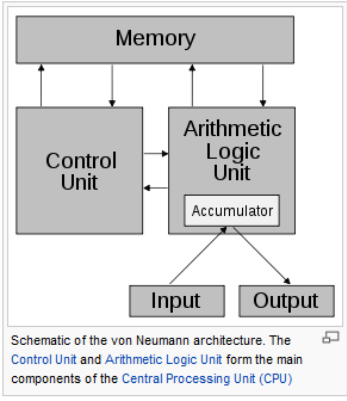
From Wikipedia

Modern Computer Hardware

Introduction Series Part I

CPU = CU+ALU

Performs and coordinates computation



Schematic of the von Neumann architecture. The Control Unit and Arithmetic Logic Unit form the main components of the Central Processing Unit (CPU)

Memory:

RAM, ROM, HD, other disks, memory sticks

I/O:

KB/Monitor Mouse, track ball, printer, network connections

Bus:

Set of wires transport data

From Wikipedia

Modern Computer Hardware

Introduction Series Part I

A modern computer consists of six subsystems:

- Central Processing Unit
- Internal Memory
- Network Connection
- Auxiliary I/O Devices
- Auxiliary Storage Devices
- User Interfaces

Computer Software

Introduction Series Part I

Software gives functionality to the hardware

Computer only understands bits (**bi**nary digi**t,** 0 or 1)

Byte is 8 adjacent bits (measures storage)

Computer Software processes complex patterns of 0s and 1s and transforms them to be viewed as text, images & video

These are meaningful instructions to the computer:

000000 00001 00010 00110 00000 100000
100011 00011 01000 00000 00001 000100

Computer Software

Introduction Series Part I

Types of software:

System Software: Is written to support basic operations of the computer, which allows users to interact with it.
For example: OS, Compilers, Communications Protocols

Application Software: Is written to support specialized tasks for users.
For example: UI Systems, Database systems, Spreadsheets and many other applications we write to solve day today needs

Computer Software

Introduction Series Part I

Types of data the computer needs to support:

Integers
Floating point
Characters
Strings
Photos
Sound
Video

But, the computer only understands 0's and 1's
??

Computer Software

Introduction Series Part I

Need a binary representation of information:

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

How do you convert 1111 in decimal?

Binary Decimal

$1\ 1\ 1\ 1 = 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$

$3\ \ 2\ \ 1\ \ 0$ Position of digits

$= 8 + 4 + 2 + 1$

$= 15$

This is similar to what you would do for decimals…

$4\ 3\ 6\ 5 = 4 * 10^3 + 3 * 10^2 + 6 * 10^1 + 5 * 10^0$

$= 4000 + 300 + 60 + 5$

$= 4365$

Computer Software

Introduction Series Part I

How about other numbering systems? For example, Octal and Hexadecimal?

From Wiki

$0_{hex} = 0_{dec} = 0_{oct}$	0	0	0	0
$1_{hex} = 1_{dec} = 1_{oct}$	0	0	0	1
$2_{hex} = 2_{dec} = 2_{oct}$	0	0	1	0
$3_{hex} = 3_{dec} = 3_{oct}$	0	0	1	1
$4_{hex} = 4_{dec} = 4_{oct}$	0	1	0	0
$5_{hex} = 5_{dec} = 5_{oct}$	0	1	0	1
$6_{hex} = 6_{dec} = 6_{oct}$	0	1	1	0
$7_{hex} = 7_{dec} = 7_{oct}$	0	1	1	1
$8_{hex} = 8_{dec} = 10_{oct}$	1	0	0	0
$9_{hex} = 9_{dec} = 11_{oct}$	1	0	0	1
$A_{hex} = 10_{dec} = 12_{oct}$	1	0	1	0
$B_{hex} = 11_{dec} = 13_{oct}$	1	0	1	1
$C_{hex} = 12_{dec} = 14_{oct}$	1	1	0	0
$D_{hex} = 13_{dec} = 15_{oct}$	1	1	0	1
$E_{hex} = 14_{dec} = 16_{oct}$	1	1	1	0
$F_{hex} = 15_{dec} = 17_{oct}$	1	1	1	1

Computer Software

Introduction Series Part I

How about characters?

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

All 128 ASCII characters including non-printable characters (represented by their abbreviation).
The 95 ASCII graphic characters are numbered from 0x20 to 0x7E (32 to 126 decimal). The space character is considered a non-printing graphic.[1]

How about Instructions?

From Wiki

Computer Software

Introduction Series Part I

What is a computer program?

- Set of instructions to solve a problem
- Is interpreted and understood by a computer
- Consists of sequences of 0s and 1s
- Machine code is cumbersome to understand by humans, so we use a language that we can understand better
- A translator (compiler) converts this code to machine language code

Computer Languages

Introduction Series Part II

- **Natural Language**
  - We use for everyday conversation
  - Contextual and ambiguous
  - Don't need to understand everything
  - Semantics and Syntax
- **Formal Language**
  - Limited vocabulary
  - Single and defined meaning
  - Appropriately called as 'Context Free Language'
- **Programming Language**
  - Is application of a formal language

Computer Languages

Introduction Series Part II

- Natural Language
  - We use for everyday conversation
  - Contextual and ambiguous
  - Don't need to understand everything
  - Semantics and Syntax
- Formal Language
  - Limited vocabulary
  - Single and defined meaning
  - Context Free Language
- Programming Language
  - Is application of a formal language

Computer Languages

Introduction Series Part II

- Natural Language
  - We use for everyday conversation
  - Contextual and ambiguous
  - Don't need to understand everything
  - Semantics and Syntax
- Formal Language
  - Limited vocabulary
  - Single and defined meaning
  - Context Free Language

- Programming Language
  - Is application of a formal language
  - Deals with Data and Procedures

Computer Languages

- Types of Programming Languages
  - **Special Purpose Language (SPL):**
    - Designed to solve specific problem. For example: SQL, LISP, Prolog, COGO, APT
  - **General Purpose Language (GPL)**
    - Designed to solve different types of problems. For example Ada, Assembly language, Basic, C, C++, Fortran, Java, Pascal, Cobol, Python, Ruby

Computer Languages

Introduction Series Part II

- **General Purpose Language (GPL)**
  - **Low-level**
    - Machine language (Generation 1 – Late 1940s – 1950s) : The only language of computer
    - Consists combination of 0s and 1s
    - Suppose you had to add two values together:
      In Algebra you would write:
      C = A + B

Computer Languages

Introduction Series Part II

- **General Purpose Language (GPL)**
  - **Low-level**
    - Machine language: The only language of computer
    - Consists combination of 0s and 1s
    - Suppose you had to add two values together:
      In Algebra you would write:
      C = A + B
    - But in Machine Language the instructions had to be in 0's and 1's
      000000 00001 00010 00110 00000 100000
      100011 00011 01000 00000 00001 000100
      Difficult and cumbersome to program, not portable
    - Assembly language
  - **High-level**

Computer Languages

Introduction Series Part II

- **General Purpose Language (GPL)**
  - **Low-level**
    - Machine language: The only language of computer
    - Assembly language: Generation 2 – Early 1950s to Present
    - Enables Machine code in words and numbers (mnemonics)
    - Example:
      - MOV AL, 61h
      - Instead of
        - 10110000 01100001
      - Means Move hex value 61 into register AL
    - Still difficult to remember and results in long codes
    - Need Assembler to translate
    - It is also processor dependent
  - **High-level**

Computer Languages

Introduction Series Part II

- **General Purpose Language (GPL)**
  - **Low-level**
    - Machine language: The only language of computer
    - Assembly language:
  - **High-level Languages are** (Generation 3 – Mid 50s to Present):
    - Abstraction by using English words
    - Natural language like and user friendly
    - Problem oriented than hardware focus
    - Portable across processors
    - Needs to be converted into Machine code
    - Ada, C, C++, Java, Fortran, Cobol, Basic

Computer Program

Introduction Series Part II

- Humans write the program (set of instructions) in high level languages like Java
- These programs, which are also known as source code, needs to be translated into machine code
- Java is one of the most popular languages of today
- To write a good program, a programmer needs to anticipate all the possible problems, and provide a solution

Software Development Process

Introduction Series Part II

- **Software Development Life Cycle (SDLC)**
  - Dictates processes to create high-quality software, which requires organization, planning and using programming conventions
- **Waterfall Model**
  - One of the methods or program development
  - Changes in one phase requires visiting previous phase

Software Development Process

Introduction Series Part II

- **Waterfall uses these Steps**
  - Define the program objectives (customer request)
  - Analysis of the program objectives
  - Design the Program
  - Write the code
  - Implementation (Compile, run, test the program)
  - Integration
  - Maintain and modify the Program
- **Cost of development**
  - Mistakes are easy and less costly to correct in earlier phases in SDLC but exponentially grows in later phases

Software Development Process

Introduction Series Part II

- **Agile Software Development Model**
  - It is team-based
  - Iterative
  - Incremental (analysis, design, implement, repeat)
  - Value driven (set priority)
  - Quality centric
  - Believes in frequent releases
  - Inspects and Adapts (changes are anticipated)
- **Agile processes**
  - Extreme Programming
  - Scrum

Software Development Paradigm

Introduction Series Part II

- **Paradigm:** *a philosophical or theoretical framework of any kind (wiki definition)*

- **Three Important Software Development Paradigm**
  - Procedural driven paradigm
  - Data driven paradigm
  - Object driven paradigm

Software Development Paradigm

Introduction Series Part II

- **Procedural Driven Paradigm**
  - Focuses on functionality desired
  - Uses top down decomposition approach
  - System architecture defined early
  - Phase wise developments
  - Changes are costly
  - Team development is messy
  - Cobol, Fortran, Basic, C, Pascal etc.

Software Development Paradigm

Introduction Series Part II

- **Data Driven Paradigm**
  - Focuses on data in question
  - Starts with entities (data) and their relation
  - System architecture defined early
  - Phase wise developments
  - Changes are costly
  - Team development is messy
  - PL-SQL etc.

Software Development Paradigm

Introduction Series Part II

- **Object Driven Paradigm**
  - Focuses on objects (nouns) in question
  - Starts with model of the application
  - System architecture is object relationship
  - Iterative process with stepwise refinement
  - Incremental development
  - Java, C++, Smalltalk etc.

Software Development Paradigm

Introduction Series Part II

- **Object Driven Paradigm**
  - Objects talk to each other by sending messages

Tax Payer object tells H&R Block Object to do his/her taxes for 2011

HRBlock.DoMyTax(2011Income, 2011Deduction)
<object>.<message>(<parameter>)

Introducing Java

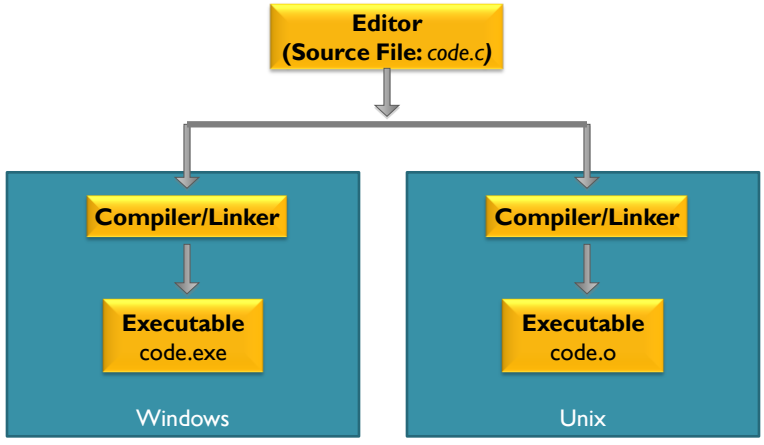
Introduction Series Part III

- **Fastest growing computer language**
- **Newest OOP language – OOP evolved!**
- **Ideal for distributed applications:**
  - Robust security
  - Better memory management
  - Programs are portable in different OS
  - Supports threads
  - Resembles C++, most robust industrial strength language
- **It is an interpreted language, so it might run slow**

Introducing Java

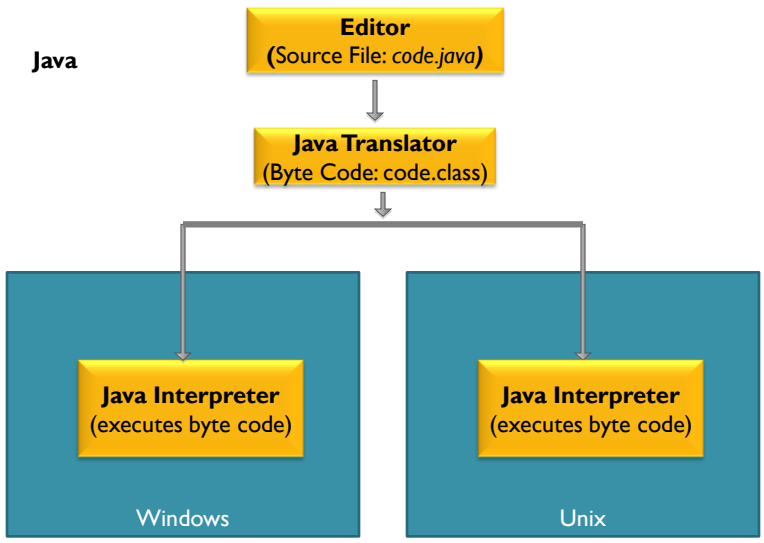
Introduction Series Part III

Other programming languages



Introducing Java

Introduction Series Part III



Introducing Java

Introduction Series Part III

- **Java Byte Code:** Translators (java compilers) translate Java code into **java byte code** which is a pseudo machine language code. It is OS independent.
- **Java Virtual Machine (JVM):** Java byte code is interpreted and run by a **interpreter** of a OS called JVM
- **JVM:** Is really like a computer within a computer. So, program runs more slowly.
- **JIT**: Just-In-Time compilation speeds up

Introducing Java

Introduction Series Part III

• **Java Virtual Machine (JVM)** has many advantages also:

- Portability: Runs in any OS
- Applet: Small already translated byte code. Runs in a embedded JVM in browser

Introducing Java

Introduction Series Part III

Java consists of three important components:

1. **The Java language:**
   a. The Java language defines the syntax and semantics of the Java programming language.
2. **The Java Virtual Machine (JVM)**
   The JVM executes Java byte code., you produce by compiling Java code
3. **The Java API (Application Programming Interface)**
   The Java API is the set of classes included with the Java Development Environment.

Introducing Java

Introduction Series Part III

Before you compile and run this code, you need to setup the Java environment in your machine

```
public class FirstJavaHello {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Helo World and Students of Java!");
	}

}
```

Introducing Java

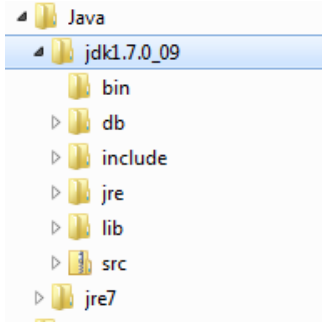
Introduction Series Part III

Many choices of Java environment today.

- Download and install JDK from Sun/Oracle

http://www.oracle.com/technetwork/java/javase/downloads/index.html

- Use notepad to write code or use IDE like eclipse



Introducing Java

Introduction Series Part III

Configure Search Path:

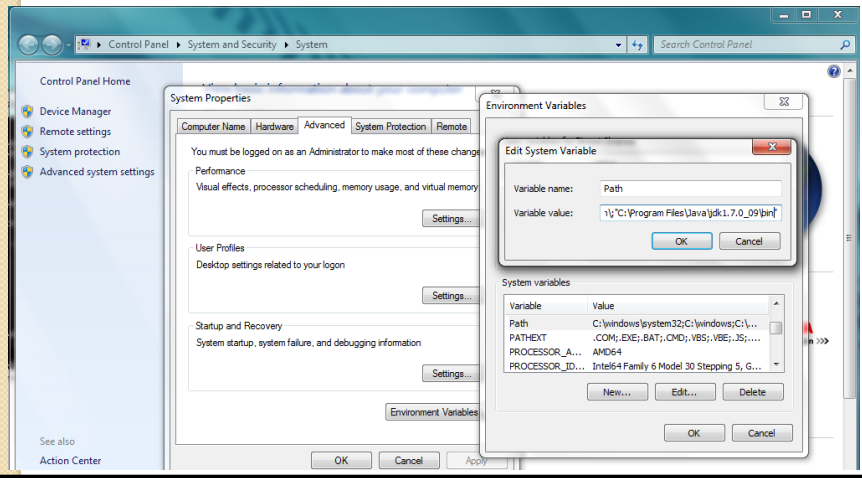
- List of directories OS searches for executable file
- PATH is a command to check current path
- Add Java JDK path in the list

  PATH *C:\Windows;C:\jdk\bin*
- Case insensitive. Use " " for name with spaces

Introducing Java

Introduction Series Part III

Configure Search Path:



Introducing Java

Introduction Series Part III

Configure CLASSPATH:

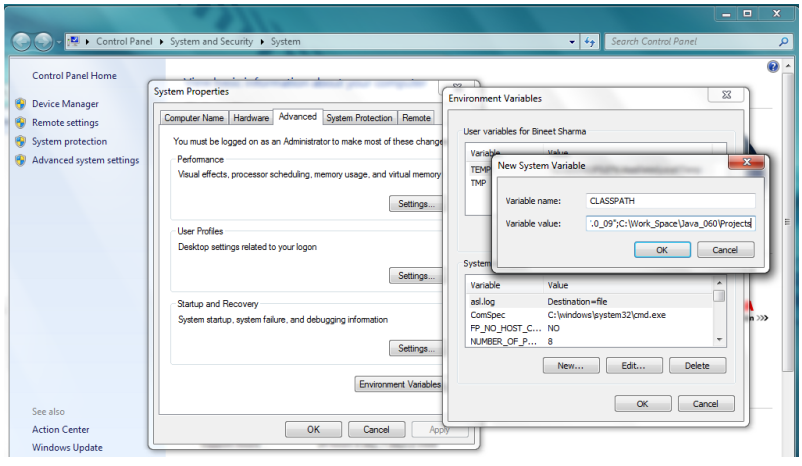
- List of dir Java system will search for its components
- Find/Edit CLASSPATH with SET command

  SET *CLASSPATH=.;C:\jdk\;C:\MyWorkSpace*
- Case sensitive. Use "" for name with spaces

Introducing Java

Introduction Series Part III

Configure CLASSPATH:



Introducing Java

Introduction Series Part III

```
public class FirstJavaHello {
	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Helo World and Students of Java!");
	}
}
```

```
C:\Work_Space\Java_060\Projects>dir
Volume in drive C is TI105835W0N
Volume Serial Number is 8EB7-BEA0

Directory of C:\Work_Space\Java_060\Projects

1/04/2012  10:00 PM    <DIR>          .
1/04/2012  10:00 PM    <DIR>          ..
1/04/2012  10:10 PM               446 FirstHello.class
1/04/2012  09:58 PM               204 FirstHello.java
               2 File(s)            650 bytes
               2 Dir(s)  427,954,405,376 bytes free

C:\Work_Space\Java_060\Projects>javac FirstHello.java

C:\Work_Space\Java_060\Projects>java FirstHello
Helo World and Students of Java!

C:\Work_Space\Java_060\Projects>
```

Introducing Java

Introduction Series Part III

Compiling and running following code:

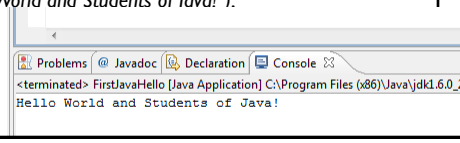
Compile *with: javac FirstJavaHello.java*

Run *with: java FirstJavaHello*

```
public class FirstJavaHello {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Helo World and Students of Java!");
	}

}
```



Vocabulary We Used

Introduction Series

- ✓ CLASSPATH
- ✓ Eclipse
- ✓ Interpreter
- ✓ Java
- ✓ Java Development Kit
- ✓ Java Virtual Machine (JVM)
- ✓ Just In Time Compiler (JIT)
- ✓ Object Oriented Programming
- ✓ Source Code
- ✓ Translator

Summary

Introduction Series

- **Computer Components**
  - Hardware and Software
  - Languages: Natural, Formal, Programming
  - Computer Programs
- **Software Development Process**
  - Software Development Life Cycle
  - Software Development Paradigm
  - Object Oriented Programming Methods
- **Introducing Java**
  - Why Java? How is it used?
  - First Java Program