1. Introduction

[ITP20003] Java Programming

Agenda

- Computer Basics
- The First Java Application
- Programming Basics

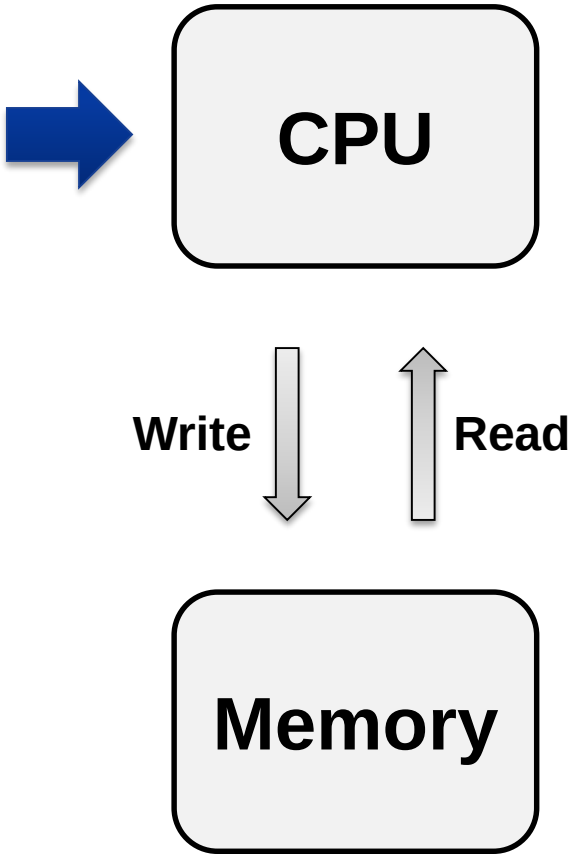
Computer

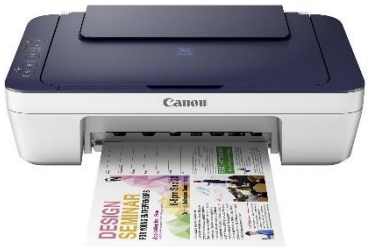
- Physical machine.

- Composed of …
  - Input devices (keyboards, mouse, camera, mic,…)
  - Output devices (monitor, printer, speaker, …)
  - Storages (HDD, SSD, flash memory, CD/DVD, …)
  - CPU, main memory, controller, …

CPU and Memory

Set of instructions
- Instruction 1
- Instruction 2
- Instruction 3
- Instruction 4
- Instruction 5
- Instruction 6
- …





CPU and Memory

- CPU - carries out only very simple instructions
  - Moving data from one place in memory to another
  - Performing some basic arithmetic (+, -, …)

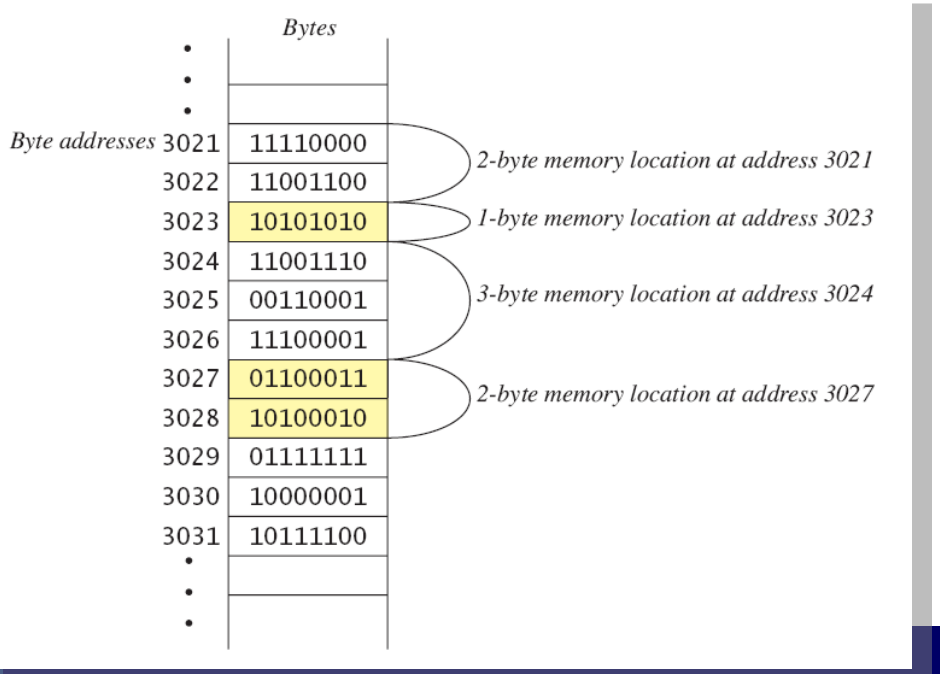
  Cf. program: a sequence of instructions to accomplish a task

- Main memory (RAM) – stores data and instructions
  - Volatile
  - Fast
  - Smaller and more expensive than auxiliary memory
  - The only storage CPU can access directly.

CPU: Central processing unit

Main Memory

- Main memory consists of a long list of numbered **bytes**.
  - All kinds of data are stored as a series of bits or bytes.
- The location of a byte is called its **address.**
  - The address of other memory unit, i.e. WORD(2bytes) or DWORD(4bytes), is the address of the starting byte.



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	TAB	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	{	|	}	~	

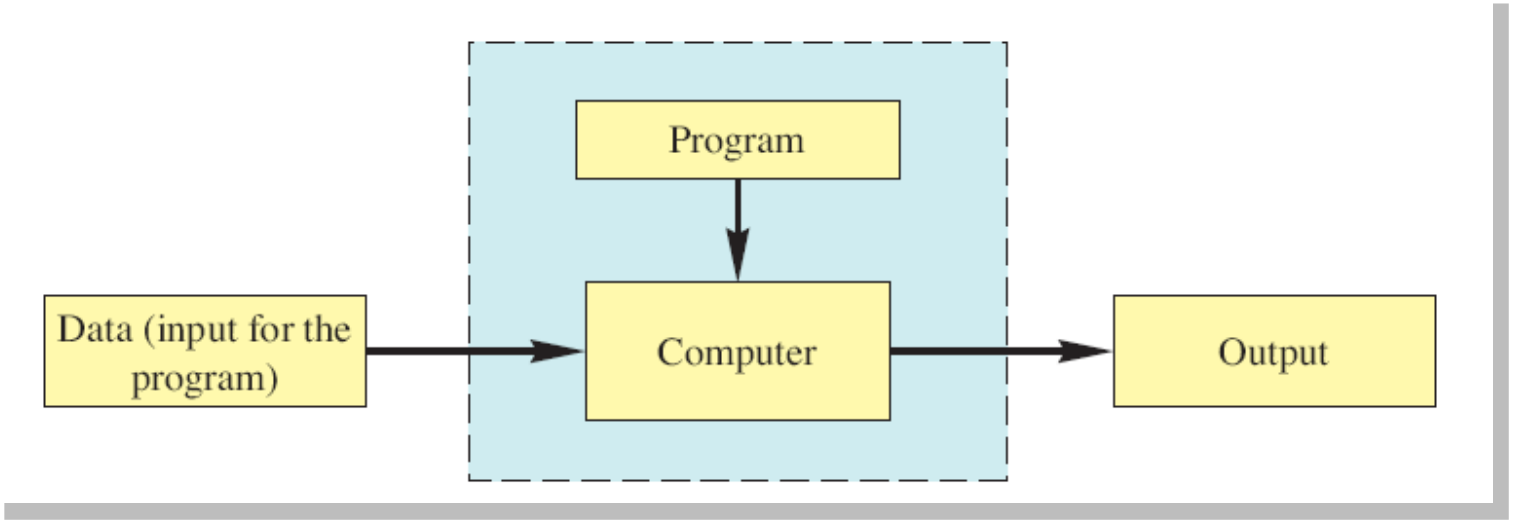
- Each character is represented by 7 bits
  - 0x00~0x1f: control characters
  - 0x20~0x7f: printable characters

    Ex) ‘a’ = 0x61, ‘0’ = 0x30, ‘-’ = 0x2D

Programs

- Program: a sequence of instructions for a computer to follow.

- Execution of program
  - Program is executed by computer (+ OS)
  - Program takes input and produces output



Programming Languages

- Primitive(low-level) programming languages
  - Machine language - a sequence of machine instructions
    - Machine instruction: primitive instructions CPU can run.
  - Assembly language – a sequence of assembly instruction
    - Assembly instruction: **symbolic** representation of machine instruction
    - **Needs translation** into machine language (assembler)

- High-level programming languages
  - Human-friendly language to describe the things the computer should do.
  - Only for human (cannot be executed on computer)

    ➔ **Needs translation** into machine language code. (interpreter/compiler)

Assembly language

A low-level programming language. There is a very strong (generally one-to-one) correspondence between the language and the machine code instructions.

Machine code bytes	Assembly language statements
	`foo:`
`B8 22 11 00 FF`	`movl $0xFF001122, %eax`
`01 CA`	`addl %ecx, %edx`
`31 F6`	`xorl %esi, %esi`
`53`	`pushl %ebx`
`8B 5C 24 04`	`movl 4(%esp), %ebx`
`8D 34 48`	`leal (%eax,%ecx,2), %esi`
`39 C3`	`cmpl %eax, %ebx`
`72 EB`	`jnae foo`
`C3`	`retl`

Instruction stream

```
B8 22 11 00 FF 01 CA 31 F6 53 8B 5C 24
04 8D 34 48 39 C3 72 EB C3
```

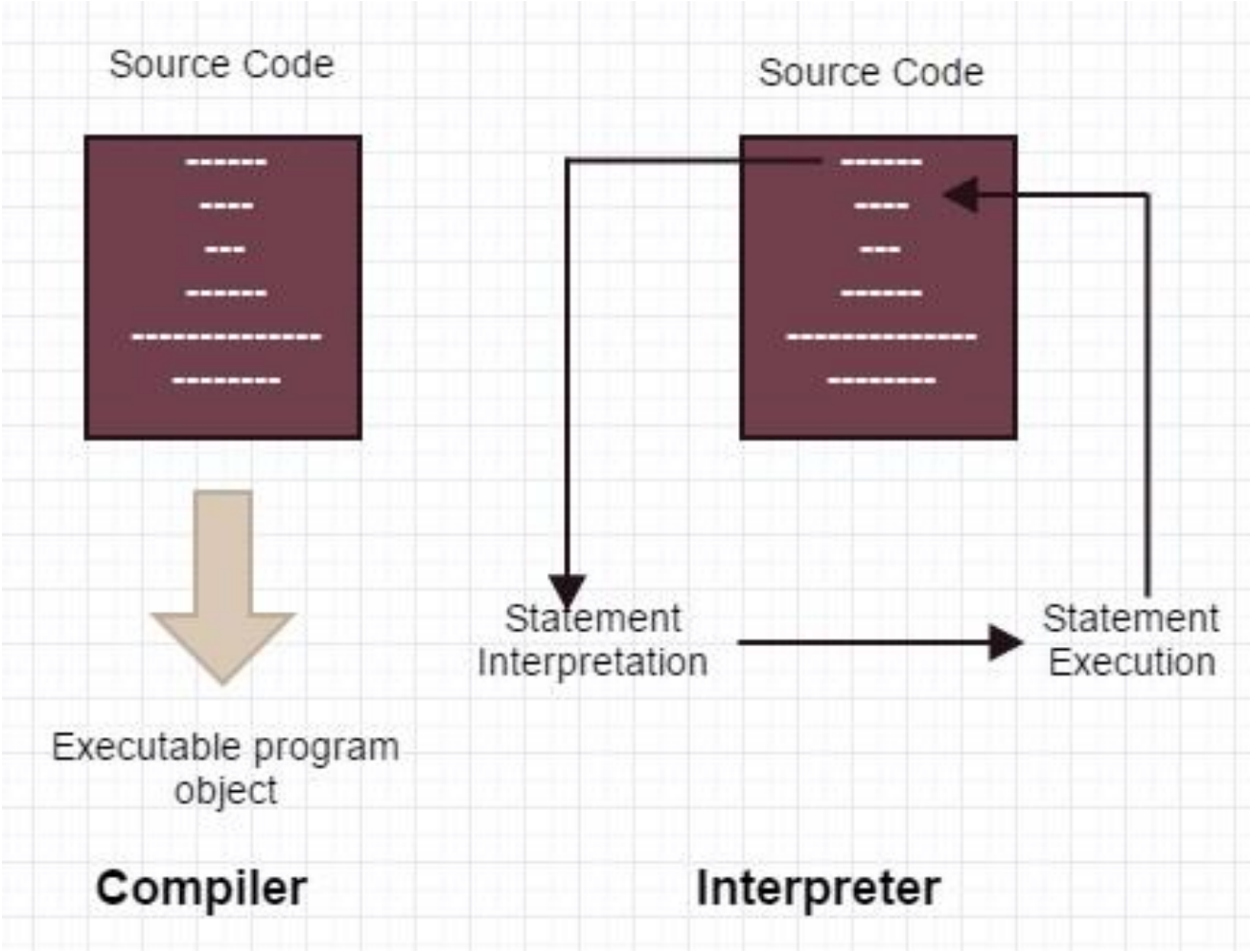

https://www.nayuki.io/page/a-fundamental-introduction-to-x86-assembly-programming

Interpreter and Compiler

- **Interpreter** - translates and executes each command alternatively
  - Translates every time the program runs.
  - Interactive

- **Compiler** - translates the whole (or a part of) program into **machine code** (exceptions: Java, C#, …)
  - Compile once execute often.
  - Fast

Interpreter and Compiler

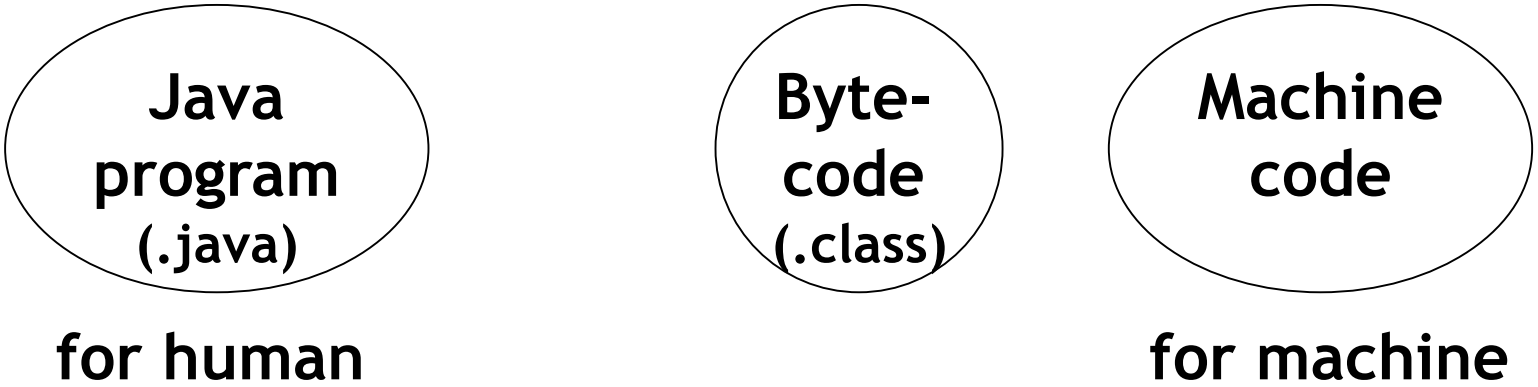


http://www.csestack.org/difference-between-compiler-and-interpreter/

Java Bytecode

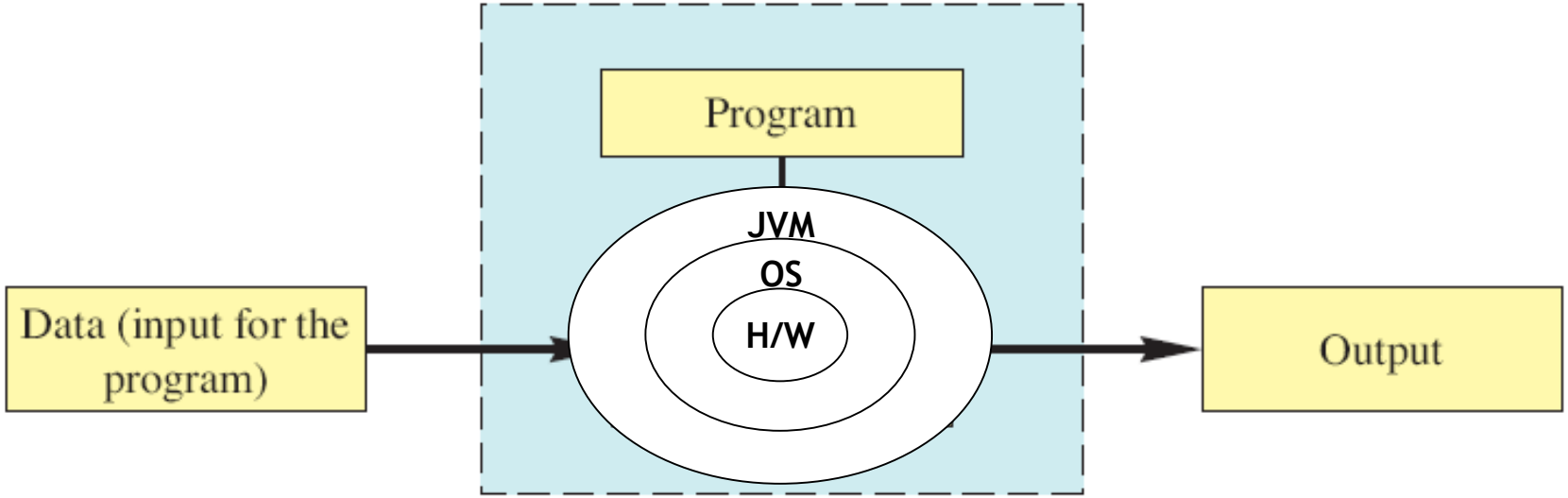
- Java compiler translates Java program into bytecode rather than machine language.

- Bytecode: machine language of a hypothetical computer known as a virtual machine, called JVM.
  - **Intermediate form** between Java program and machine code.
  - Easy to interpret

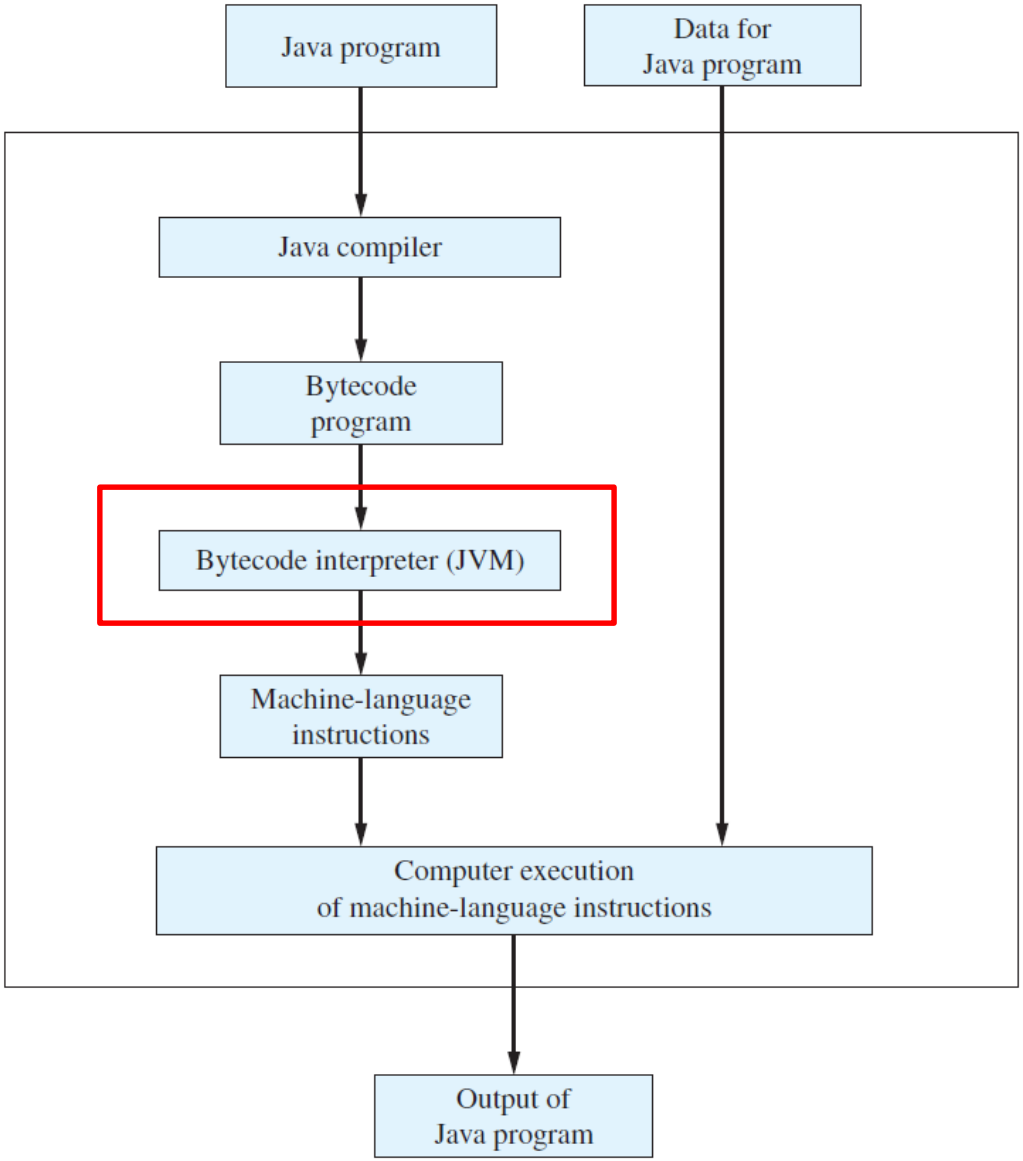


Java Virtual Machine (JVM)

- JVM interprets bytecode (translation + execution)
- JVM provides platform-independent environment.
  - There exists JVMs for various H/W's and OS's
  - Java bytecode can run on any JVM.



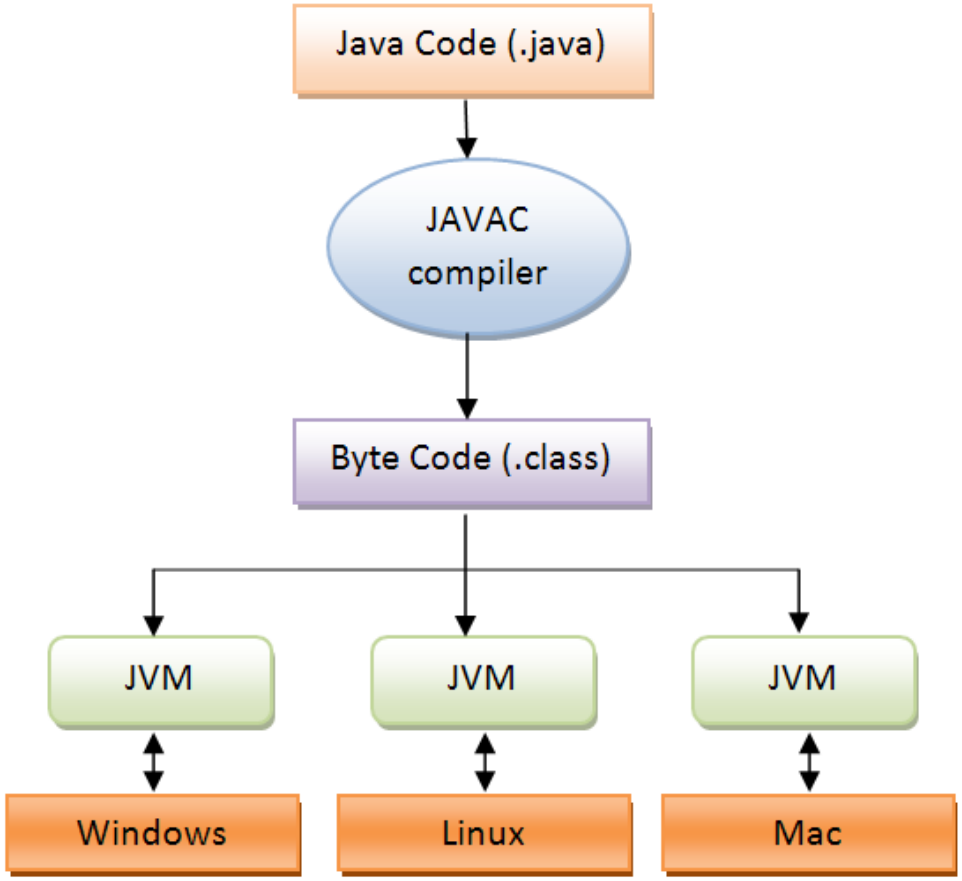
Compiling and Running Java



Java Virtual Machine (JVM)

- JVM provides great **portability**.

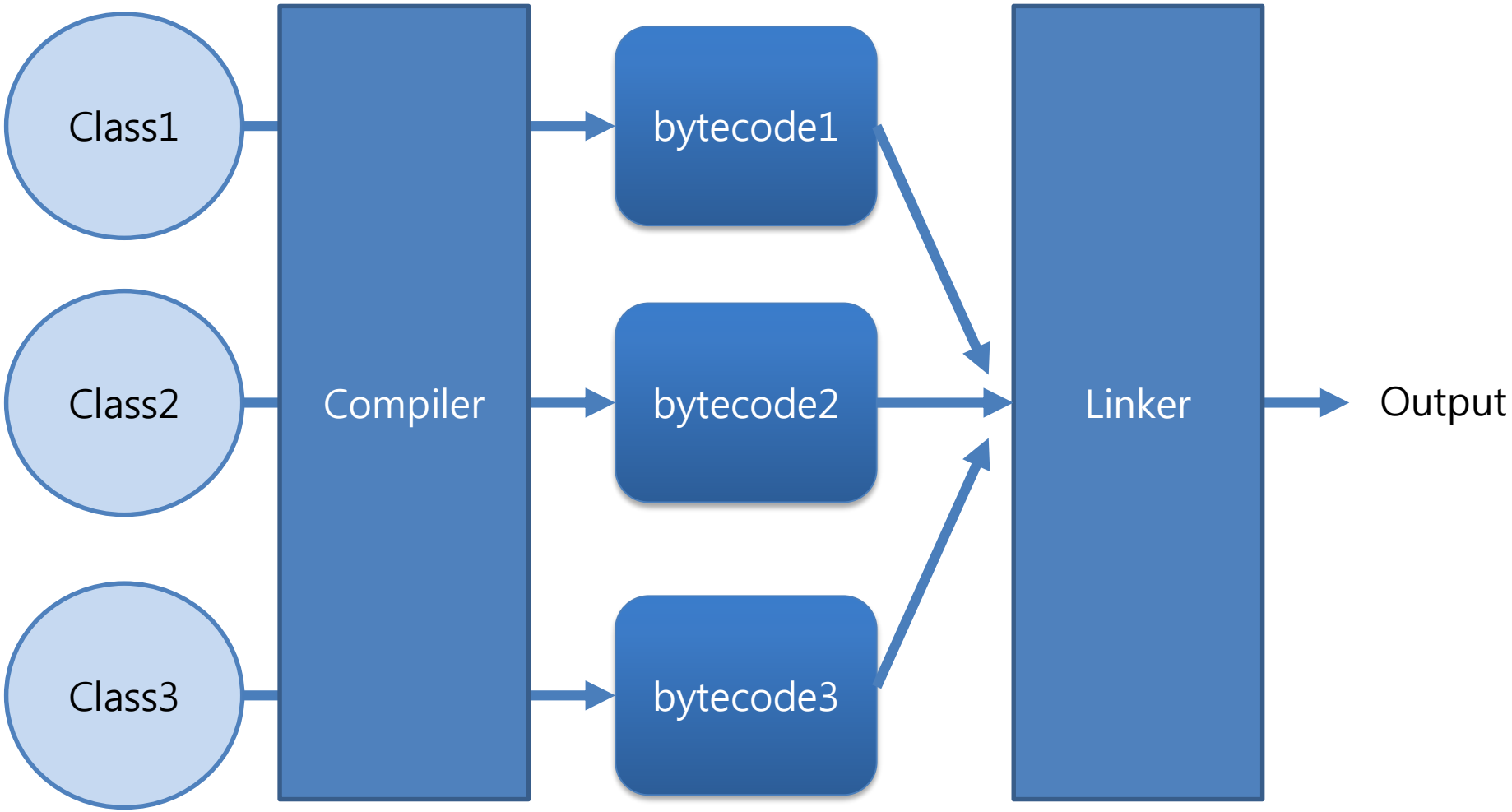
  "Compile once, run everywhere!"



Class Loader

- A Java program is seldom written as one piece of code all in one file.
- Instead, it typically consists of different pieces, known as classes.
- Class Loader connects the classes to run the program.
  - This connecting is typically done automatically.
  - Class loader corresponds to the linker of other programming language.

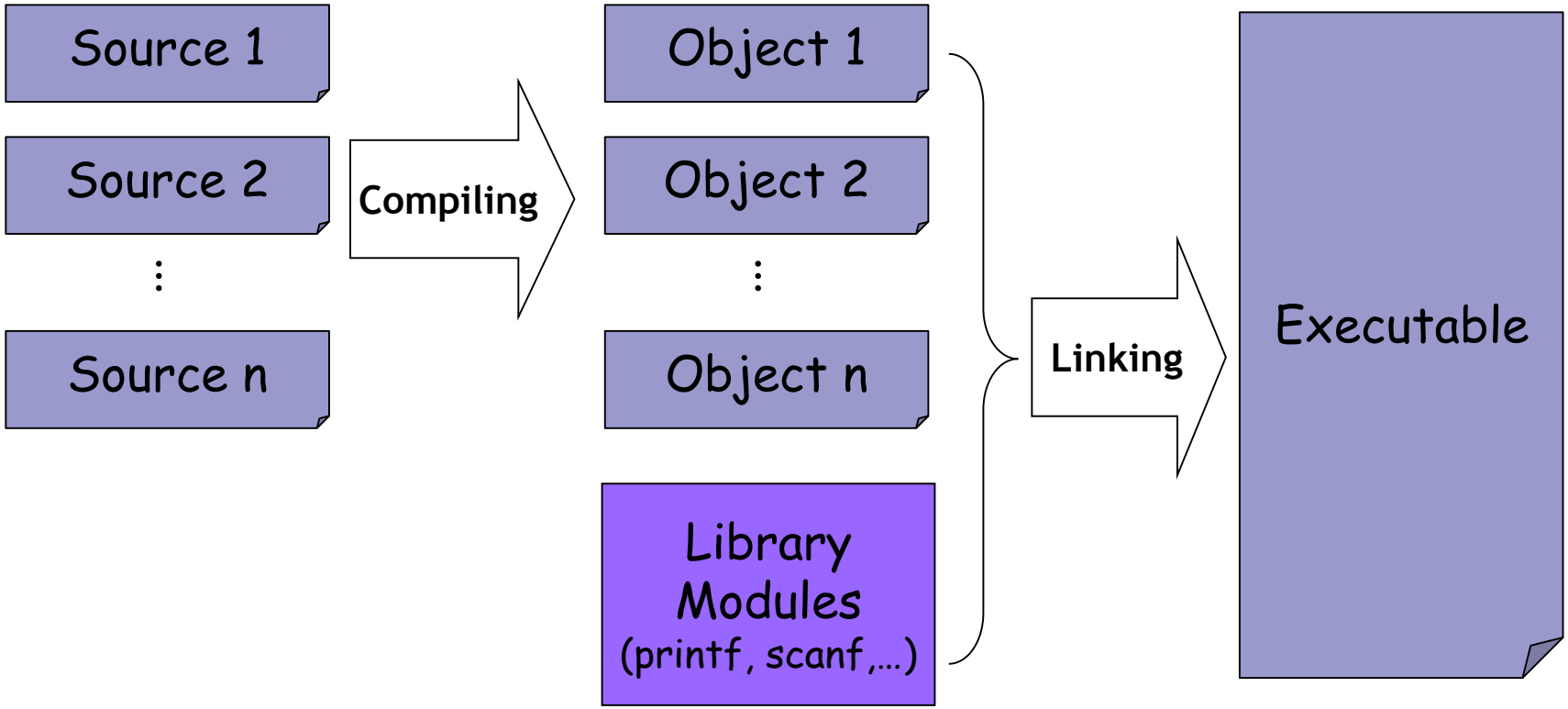
Class loader (linker)



Creating and Running C Programs

- Link
  - Integrating objects and library modules required to execute

Notice! a program can be distributed in multiple source files.



Agenda

- Computer Basics
- **The First Java Application**
- Programming Basics

Applications and Applets

- Application: regular program.
  - Run on your computer
    - H/W + OS + VM
- Applet
  - Sent to another location on the Internet and run there.
    - Web browser + VM

The First Java Application Program

```
import java.util.Scanner;
public class FirstProgram
{
    public static void main (String [] args)
    {
        System.out.println ("Hello out there.");
        System.out.println ("I will add two numbers for you.");
        System.out.println ("Enter two whole numbers on a line:");
        int n1, n2;
        Scanner keyboard = new Scanner (System.in);
        n1 = keyboard.nextInt ();
        n2 = keyboard.nextInt ();
        System.out.println ("The sum of those two numbers is");
        System.out.println (n1 + n2);
    }
}
```

The First Java Application Program

- Result

```
Hello out there.
I will add two numbers for you.
Enter two whole numbers on a line:
12 30
The sum of those two numbers is
42
```

The First Java Application Program

```
import java.util.Scanner;

public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        System.out.println("Enter two whole numbers on a line:");

        int n1, n2;

        Scanner keyboard = new Scanner(System.in);
        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();

        System.out.print1n("The sum of those two numbers is");
        System.out.print1n(n1 + n2);
    }
}
```

Gets the `Scanner` *class from the package (library)* `java.util`

Name of the class—your choice

Sends output to screen

Says that `n1` *and* `n2` *are variables that hold integers (whole numbers)*

Readies the program for keyboard input

Reads one whole number from the keyboard

The First Java Application Program

- import java.util.Scanner;
  - Tells the compiler that this program uses the class Scanner.

- class FirstProgram // see OOP

```
public class FirstProgram
{
. . .
}
```

- The main method

```
public static void main(String[] args)
{
. . .
}
```

The First Java Application Program

- System.out.println()
  - Displays what is shown in parentheses
  - System.out is an object used to send output to the screen
  - println is the method that performs this action for the object System.out.

- int n1, n2; // variable declaration
  - variable: a memory space with a name to store a piece of data.
  - int: data type (integer)
  - n1, n2: variable names

The First Java Application Program

- Scanner keyboard = new Scanner(System.in);
  - Prepares to read from the keyboard
  - System.in is an object used to read input to the keyboard

- n1 = keyboard.nextInt();
- n2 = keyboard.nextInt();
  - Reads integer numbers from the keyboard

Writing a Java Program

- A Java program is composed of smaller parts, called classes
  - List 1.1 uses three classes: FirstProgram, System, Scanner
  - Each class should be in a separate file with the same filename.

    Ex) FirstProgram.java

- Writing a Java program = writing classes
  - Design the whole program
  - Decompose it into classes
  - Implement each class

Compile and Running a Java Program

- Compile and Running with JDK (Java Development Toolkit)
  - Compiler + JRE (incl. JVM)

  cf: JRE: Java Runtime Environment (JVM + built-in classes + α)

  - Compile: javac FirstProgram.java
  - Run: java FirstProgram

  ➔ JDK should be installed, and its *bin* directory should be in PATH.

- IDE (Integrated Development Environment)
  - Editor + compiler + runtime + debugger + …

  Ex) Eclipse, NetBeans, …

  - Background compile
  - Run
    - Menu->Run->Run As->Java Application
    - Menu->Run->Run
    - CTRL-F11

Official Compiler – JDK and Eclipse

workspace - Java EE - MyProject/src/MyApp.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Project Explorer

- MyProject
  - src
    - (default package)
      - MyApp.java
  - JRE System Library [JavaSE-1.8]

MyApp.java

```
1
2  public class MyApp {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6          System.out.println("This is my first application");
7
8      }
9
10 }
11
```

Outline

- MyApp
  - main(String[]) : void

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> MyApp [Java Application] C:₩Program Files₩Java₩jdk1.8.0_121₩bin₩javaw.exe (2017. 2. 24. 오후 7:31:29)

```
This is my first application
```

Official Compiler – Repl.it

Repl.it - testbed_Java

https://repl.it/@minkyuahn/testbedJava

@minkyuahn/testbed_Java

No description

run ▶ share repl talk my repls teacher student minkyuah...

Files

Main.java

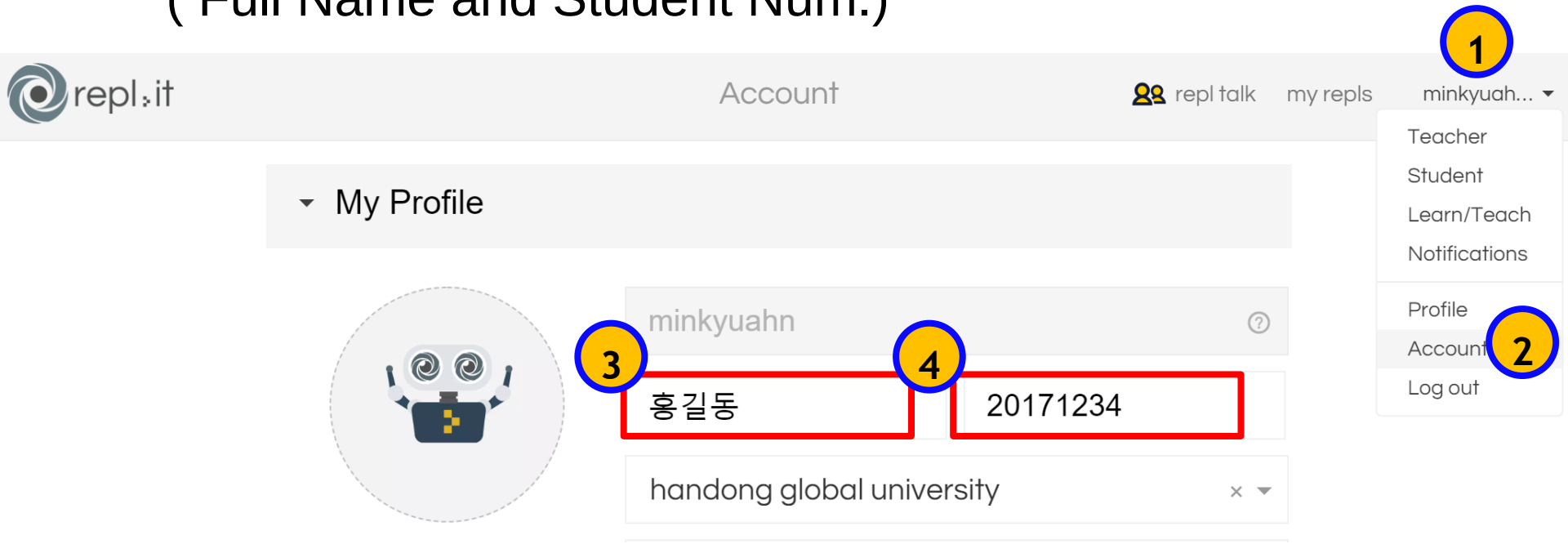
Main.java saved

```
1  import java.util.*;
2
3  public class Main {
4
5    public static void main(String[] args) {
6
7
8  }
9
```

```
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed
ode)
```

Repl.it

- Sign up – Log in – Enroll the camp
- Direct link to join the class in repl.it
  https://repl.it/classroom/invite/7I6TCnK
- Change account setting
  (Full Name and Student Num.)



[EX01] First Java Program

repl.it - [EX01] First Java Progra

https://repl.it/teacher/assignments/2850605/preview

repl.it

[EX01] First Java Program (preview)

repl talk
ai competition

my repls | teacher | student | minkyuah...

back to edit | run

```
class Main {
  public static void main(String[] args) {



  }
}
```

```
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
```

Due: Feb 28, 2019 11 : 15 am

submit

Instructions from your teacher:

Write a program that prints "Hello world!".

Agenda

- Computer Basics
- The First Java Application
- **Programming Basics**

Object-Oriented Programming

- Java is an object-oriented programming language, abbreviated OOP.
  - OOP is a technique that experienced programmers have found to be extremely helpful.

- The world is made up of objects.

  Ex) people, automobiles, buildings, …

- Object-oriented programming (OOP) treats a program as **a collection of objects** that **interact by means of actions**.

Object-Oriented Programming

- Objects, appropriately, are called **objects**.
- Actions are called **methods**.
- Objects of the same kind have the same type and belong to the same **class**.
  - Objects within a class have **a common set of methods** and **the same kinds of data**
  - But each object can have **it's own data values**.

Class, Object, and Methods

- **Class**: a type of entities

  Ex) Sonata, Genesis, Galaxy Note, i-Pad…

- **Object**: a specific entity

  Ex) my Sonata (with a specific vehicle information/plate numbers)

- **Method**: an action an object can perform

  Ex) Sonata has *go*, *stop*, *left_turn*, *right_turn*, …

- **Attribute**: component that constructs an object
  - Also called fields, member variable, data member, …

  Ex) body, engine, wheel, tire, chair, door, trunk, …

OOP Design Principles

- OOP adheres to three primary design principles:
  - Encapsulation
  - Polymorphism
  - Inheritance

Encapsulation

- The data and methods associated with any particular class are encapsulated ("put together in a capsule"), but only part of the contents is made accessible.
  - Encapsulation provides a means of using the class, but it omits the details of how the class works.

    Ex) accelerator pedal, brake pedal, steering wheel, …
  - Encapsulation often is called information hiding.

    Ex) fuel injectors, automatic braking control system, power steering pump, …

Polymorphism

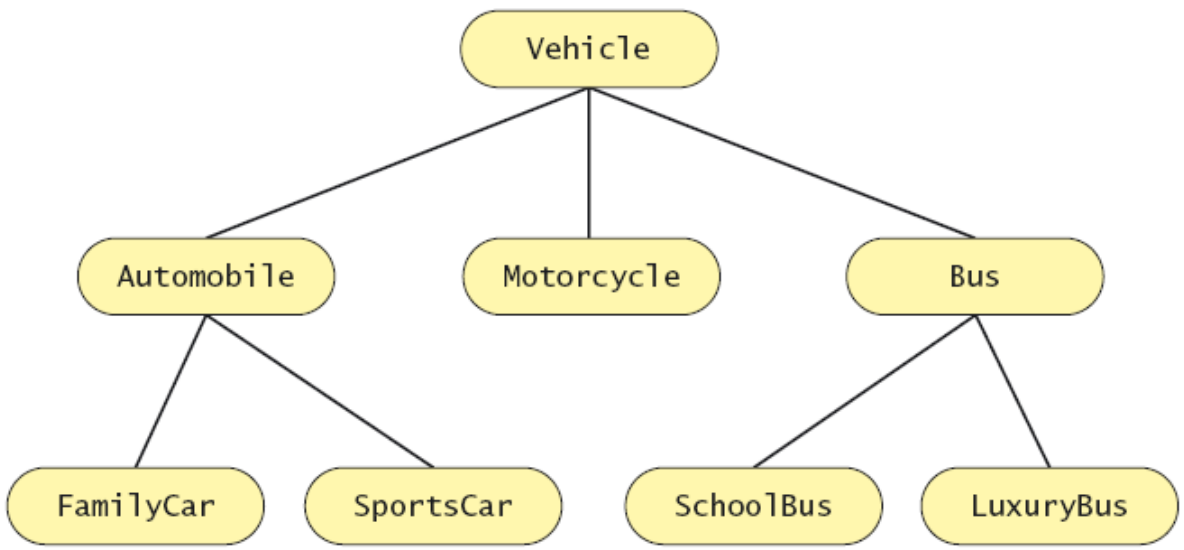
- From the Greek meaning “many forms”

- The same program instruction adapts to mean different things in different contexts.
  - A method name produces results that depend on the class of the object that used the method.

  Ex) ‘go’ method of an automobile vs. ‘go’ method of an airplane.

Inheritance

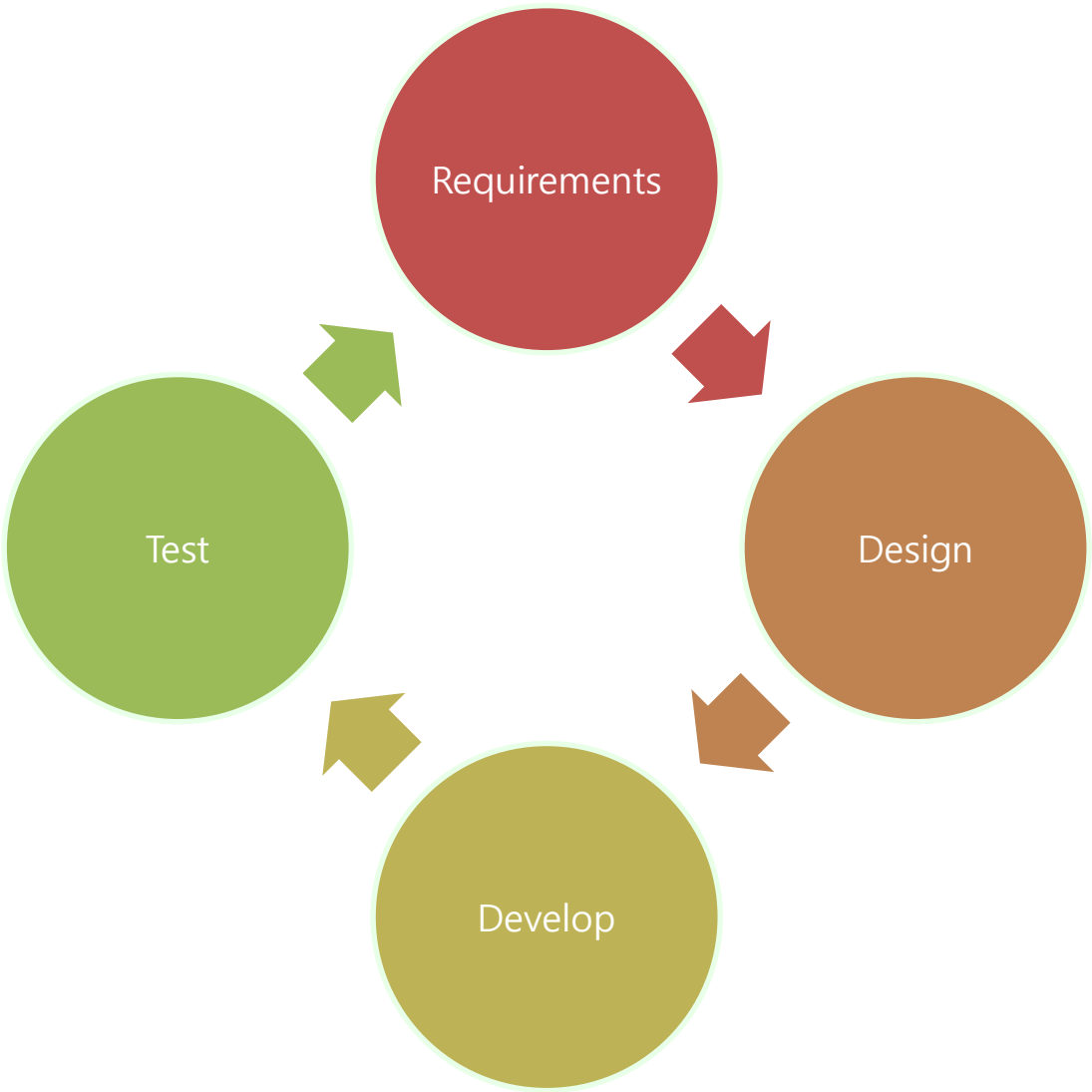
- Classes can be organized using inheritance.
  - ‘is a’ relation
- A class at lower levels inherits all the characteristics of classes above it in the hierarchy.
  - Inherited characteristics do not need to be repeated.
  - New characteristics are added.



Inheritance in Java

- Used to organize classes
- New characteristics are added.

Software development process



Algorithms

- An **algorithm** describes a means of performing an action.
  - Algorithm = a series of actions
    cf. program = a series of instructions (or commands)
  - An abstracted form of program.
  - For human, not machine

- Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.

- An algorithm must be expressed completely and precisely.

- Algorithms usually are expressed in pseudocode.

Algorithms and Pseudocode

- An algorithm is a set of directions for solving a problem.
- Pseudocode is a mixture of English and Java. When using pseudocode, you simply write each part of the algorithm in whatever language is easiest for you.

Problem **What is the summation from 1 to 100 ?**
Solution **Let's add one by one !**

Then algorithm?

```
s = 0;

From i=1 to 100

      s=s+i;

end
```

Testing and debugging

- Every program should be tested. Testing is a part of procedure of development and very important.

- Without testing, we cannot guarantee that your program works correctly.

- Bug
  Any mistakes in a program that make errors.

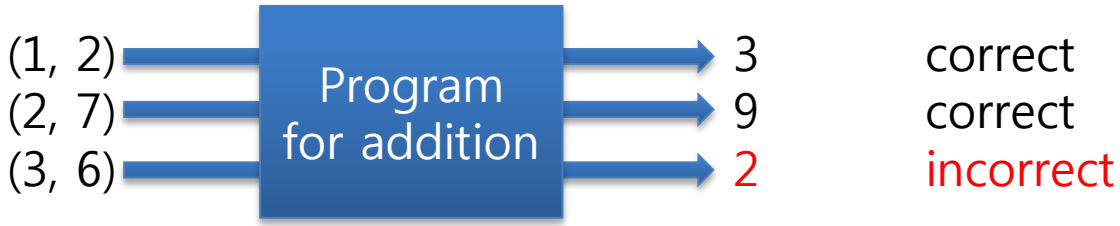
- Debugging
  The process of eliminating mistakes in your program.

Types of error

- Syntax error
  - Grammatical mistake in the program.
  - Easy to correct because the compiler says you have a syntax error.

- Runtime error
  - Error that is detected when your program is run.
  - Ex) division a number by zero

- Logic error
  - Syntactically correct but logically wrong.



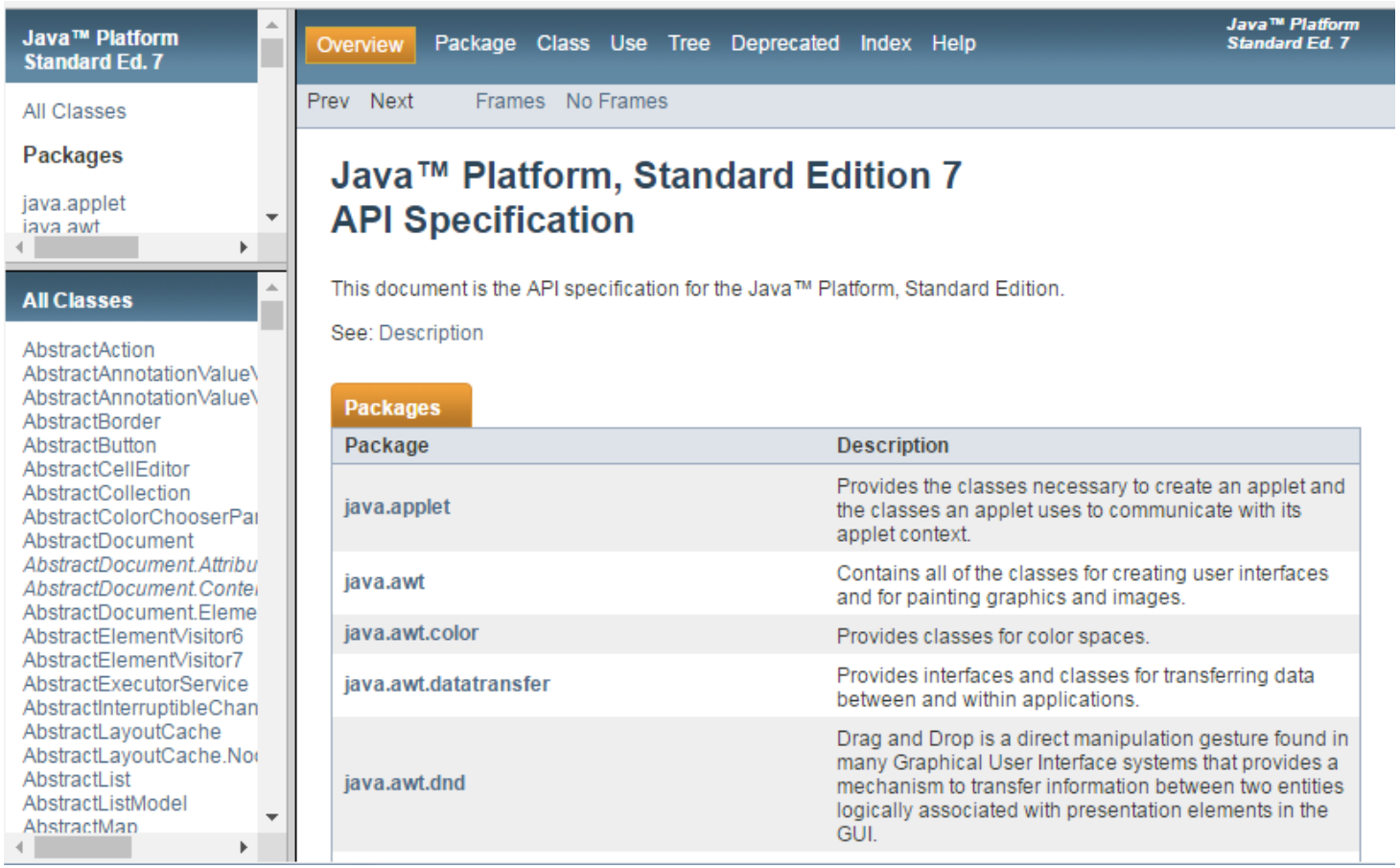
Reusable Components

- Most programs are created by combining existing components.
  - Programs NOT usually created entirely from scratch.
- Reusing components saves time and money.
- Reused components are likely to be better developed, and more reliable.
- New components should be designed to be reusable by other applications.
- Java provides many classes

  http://docs.oracle.com/javase

Java Class Library

- Java provides libraries that you can just use.
- https://docs.oracle.com/javase/7/docs/api/



Java Class Library

Java 9 is currently available.
https://docs.oracle.com/javase/9/docs/api/overview-summary.html

https://docs.oracle.com/javase/9/docs/api/overview-summary.html

Overview (Java SE 9 & JDK...

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP Java SE 9 & JDK 9

PREV NEXT FRAMES NO FRAMES ALL CLASSES SEARCH:

Java® Platform, Standard Edition & Java Development Kit
Version 9 API Specification

This document is divided into three sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with `java`.

JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

JavaFX

The JavaFX APIs define a set of user-interface controls, graphics, media, and web packages for developing rich client applications. These APIs are in modules whose names start with `javafx`.

Java SE

Module	Description
java.activation	Defines the JavaBeans Activation Framework (JAF) API.
java.base	Defines the foundational APIs of the Java SE Platform.
java.compiler	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.corba	Defines the Java binding of the OMG CORBA APIs, and the RMI-IIOP API.
java.datatransfer	Defines the API for transferring data between and within applications.
java.desktop	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.
java.instrument	Defines services that allow agents to instrument programs running on the JVM.
java.logging	Defines the Java Logging API.

90%