Java Basic

Credits: Some slides are credits to Prof. Wang Chung E.

Overview

- What is Java ?
- Features of Java
- Java and its impacts
- Java’s environment
- Programming languages ranking
- Lexical Structure & reserved words
- Java types
- Compiling a simple example
- Input/Output:
  - Conversion characters
  - Field width/precision

- Java libraries

What is Java?

Bytecode for virtual machines

Every thing is a class

RMI, Servlet, Applet

"...A **simple, pure object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded, and dynamic** language."

[Sun Microsystems, Summer 95]

Strong typing + no pointer + auto garbage collection

Java is designed to reduce many security risks.

Features of Java

- Simple
- Architecture-neutral
- Object-Oriented
- Distributed
- Compiled
- Interpreted
- Statically Typed
- Multi-Threaded
- Garbage Collected

- Portable
- High-Performance
- Robust
- Secure
- Extensible
- Well-Understood

How Will Java Change My Life?

- Get started quickly
- Write less code
- Write better code (promoting)
- Develop programs faster
- Avoid platform dependencies with 100% pure Java
- Write once, run anywhere
- Distribute software more easily

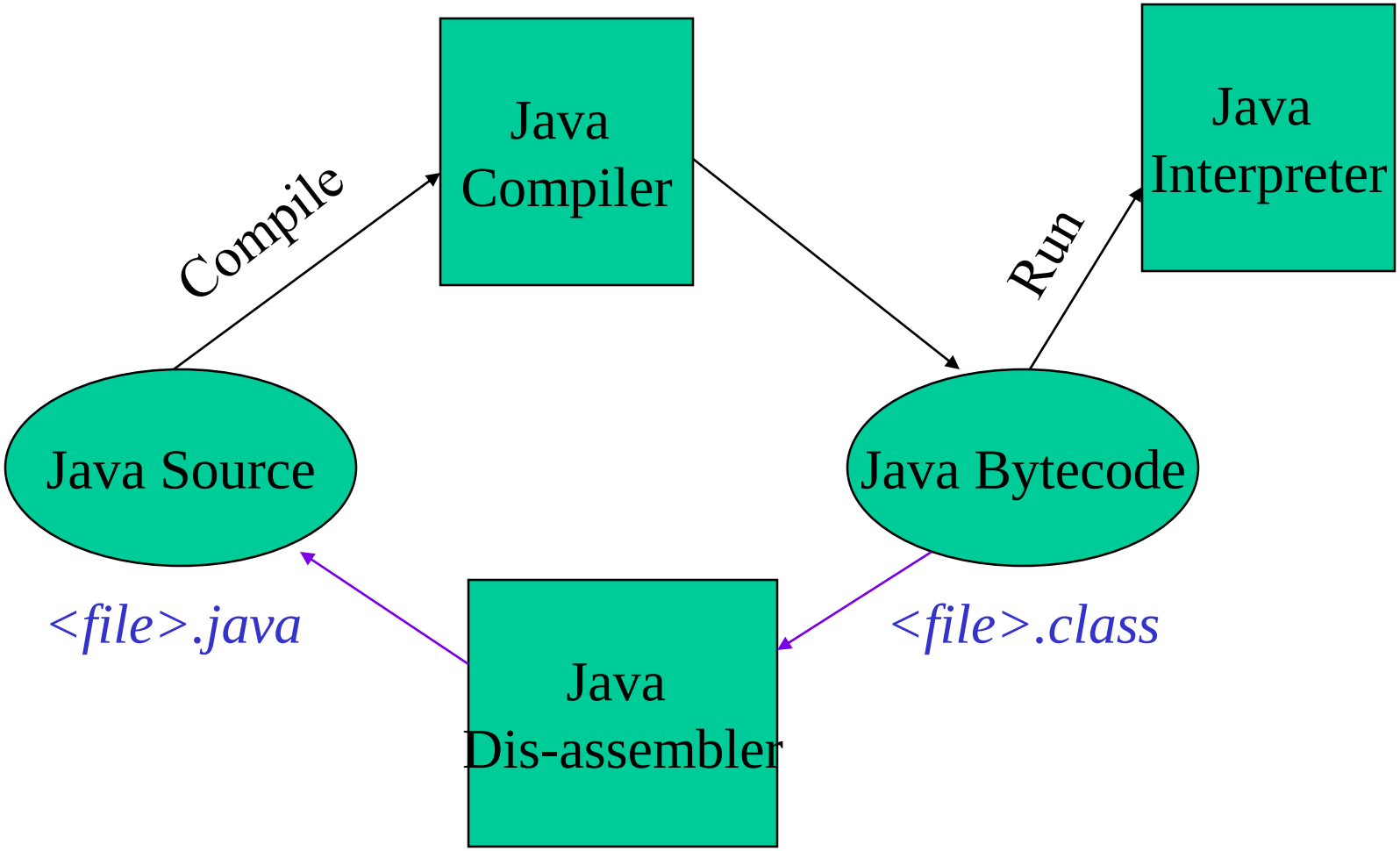
Java Applications and Java … lets

- Stand-alone Applications
  - Just like any programming language
- Applet
  - Run under a Java-Enabled Browser
- Midlet
  - Run in a Java-Enabled Mobile Phone
- Servlet
  - Run on a Java-Enabled Web Server…

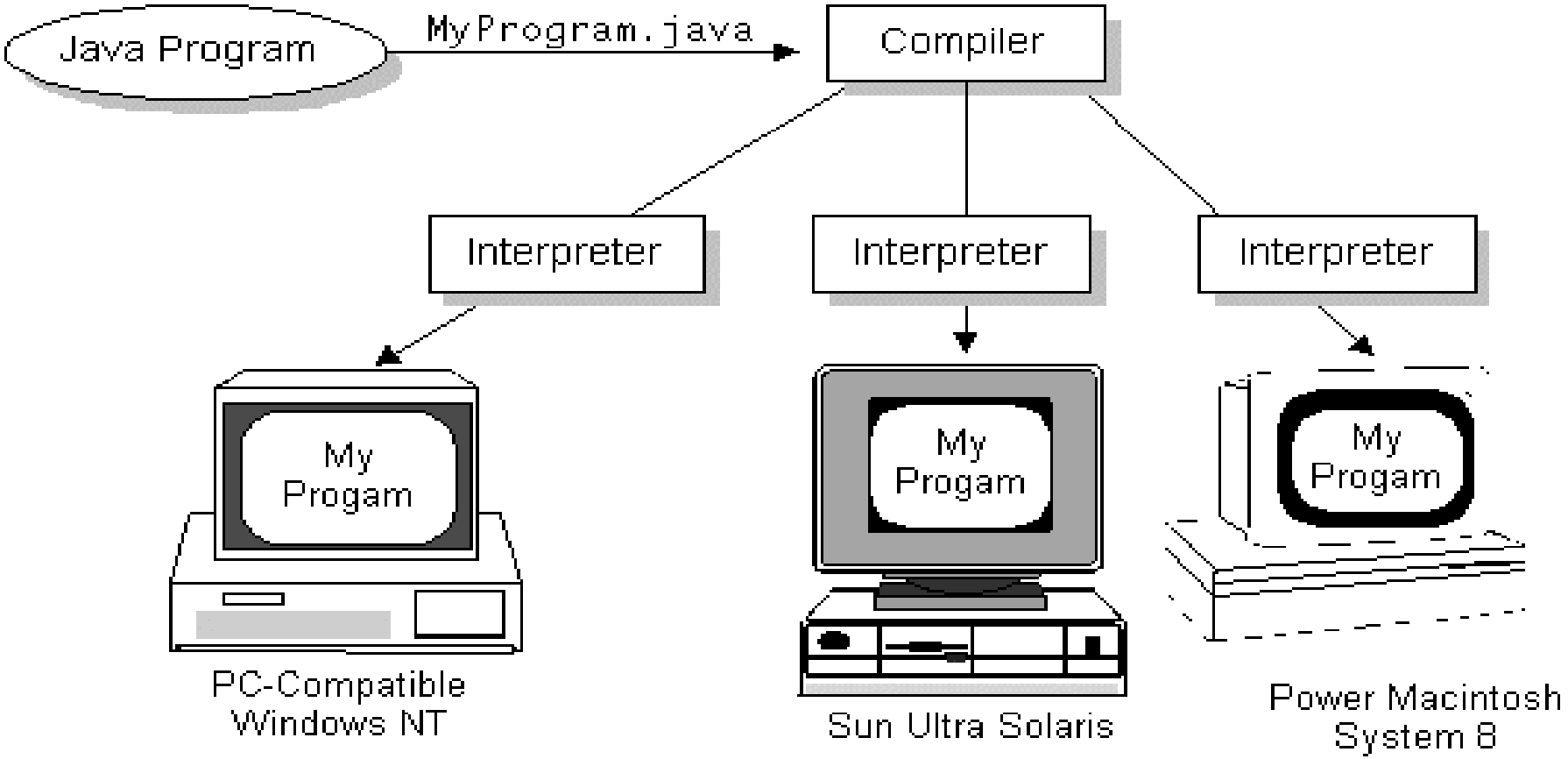
Java Developer's Kit (I)

- Java's programming environment
  - Core Java API
    - The Java API includes everything from collection classes to GUI classes. You can view "Java™ Platform, Standard Edition 8 API Specification."
  - Compiler/interpreter
  - debugger
  - dis-assembler
  - Profiler
    - provides you with a finer view of your target application execution and its resource utilization.

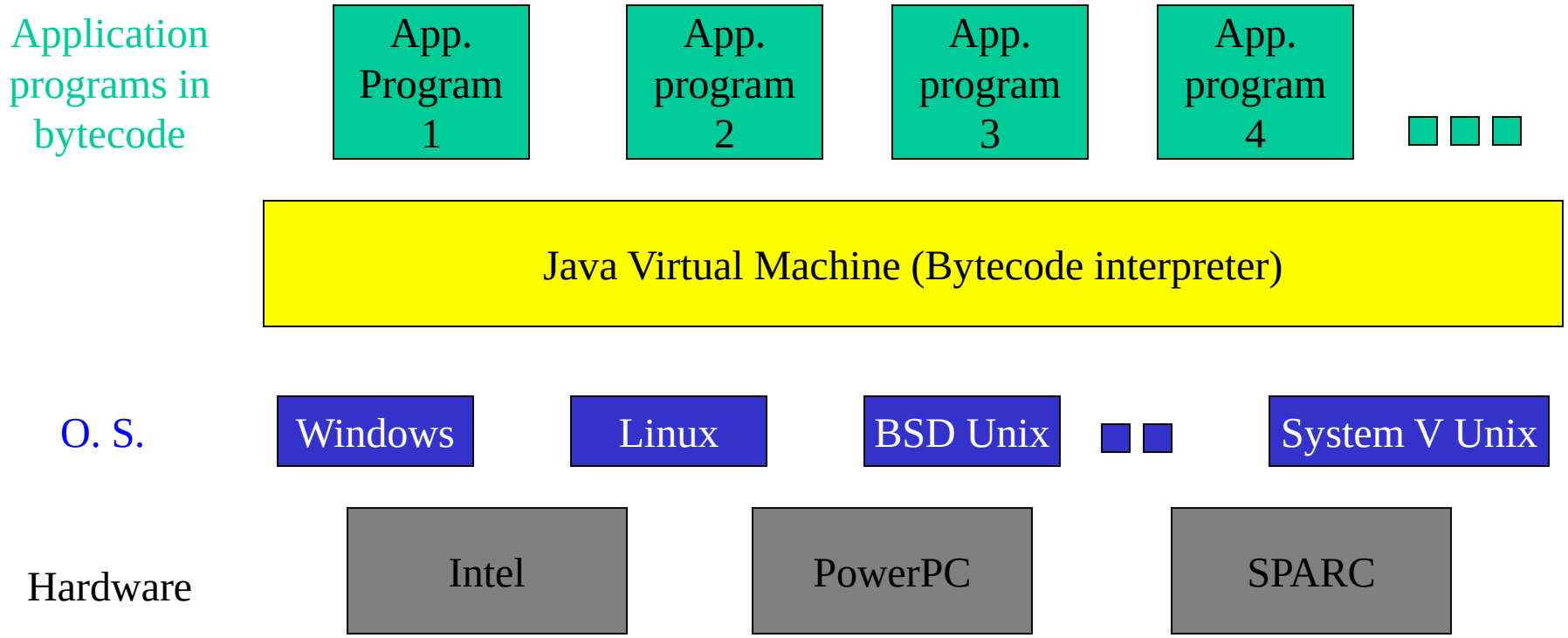
Java Developer's Kit (II)



Write Once, Run Anywhere



The Java Platform



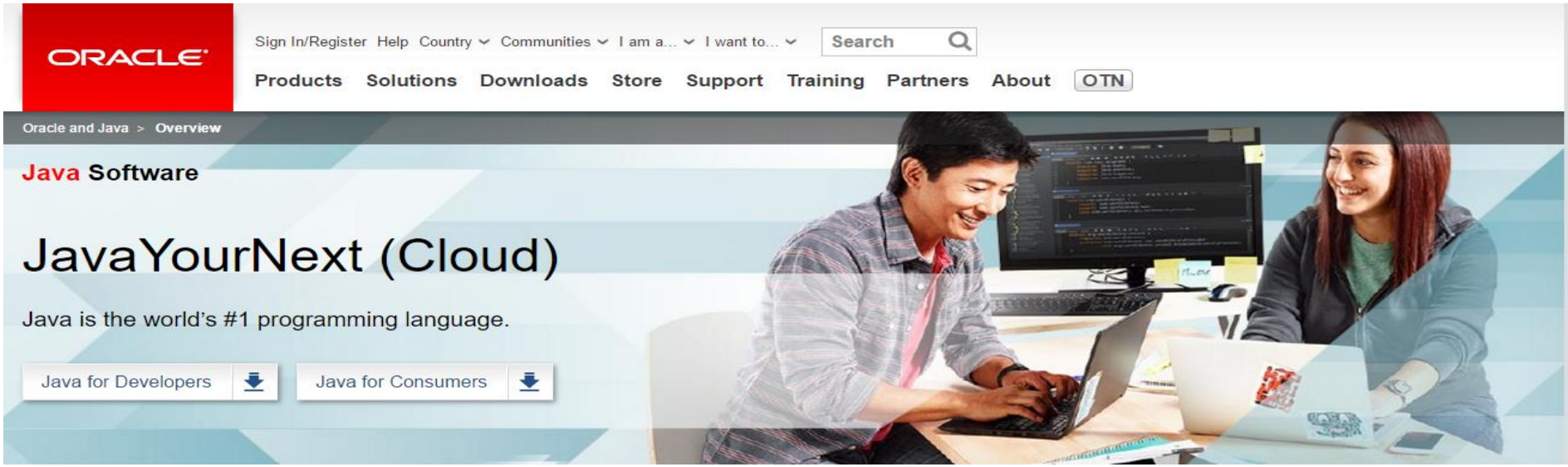
Write Once, Run Anywhere

The Java as a popular programming language

Language Rank	Types	Spectrum Ranking
1. Java		100.0
2. C		99.2
3. C++		95.5
4. Python		93.4
5. C#		92.2
6. PHP		84.6
7. Javascript		84.3
8. Ruby		78.6
9. R		74.0
10. MATLAB		72.6

Sources: http://www.tiobe.com/tiobe-index/ (languages ranking by popularity – Jan 2017)

https://www.oracle.com/java/index.html



Lexical Structure of Java

The lexical structure of a programming language is the set of elementary rules that define what are the *tokens* or basic atoms of the program. Some of the basic rules for Java are:

- Java *is* case sensitive.
- Whitespace, tabs, and newline characters are ignored except when part of string constants. They can be added as needed for readability.
- Single line comments begin with //
- Multiline comments begin with /* and end with */
- Documentary comments begin with /** and end with **/
- **Simple statements terminate in semicolons!** Make sure to *always* terminate simple statements with a semicolon.

Reserved Words

Reserved words are words that can't be used as *identifiers*. Many of them are *keywords* that have a special purpose in Java.

Java reserved keywords: (45)

abstract, **boolean**, **break**, byte, **case**, catch, **char**, **class**, **continue**, **default**, **do**, **double**, **else**, **extends**, **final**, finally, float, **for**, **if**, **implements**, **import**, **instanceof**, **int**, **interface**, **long**, native, **new**, **package**, **private**, **protected**, **public**, **return**, **short**, **static**, **super**, **switch**, synchronized, **this**, **throw**, throws, transient, **try**, **void**, volatile, **while**.

- const, goto are reserved keywords not in use.
- **null**, **true**, **false** are reserved literals in java.

Java Types

1. Primitive types
   - byte (8 bits), short (16 bits), int (32 bits), long (64 bits)
   - float (32 bits), double (64 bits)
   - char - unicode! e,g, ‘\u12ab’ (16 bits)
   - boolean (16 bits, true/false)
2. Reference types (Subtypes of Object)
   - Classes: String etc.
   - Arrays

Note: The default values for variables are 0/false/null.

Operators

+, -, *, /, <<, >>, >>>, &, |, ^, &&, ||, ... etc.

A simple Java program

```
// Hello.java
public class Hello {
	public static void main(String[] args) {
		System.out.println("Hello, World!");
	}
}
```

main() method

- java interpreter starts by calling the public class's main() method.
- main() method takes an array of String as an argument
- main() method must be declared public, static and not return a value (void).
- *Signature of the main() method can be any of these:*
  - public static void main(String args[])
  - public static void main (String [] args)
  - static public void main (String [] args)
    - note: args can instead be any valid identifier like "anything"

System.out.println

- System is a predefined class that provides access to the system
  - **System** is a final class from java.lang package.
- out is outputstream that connect to the Console
  - out is the reference variable of PrintStream class and a static member of System class
- println displays the string which passed to it
  - **println** is a method of PrintStream class.

Online resources –

The official Java site :

https://www.oracle.com/java/index.html

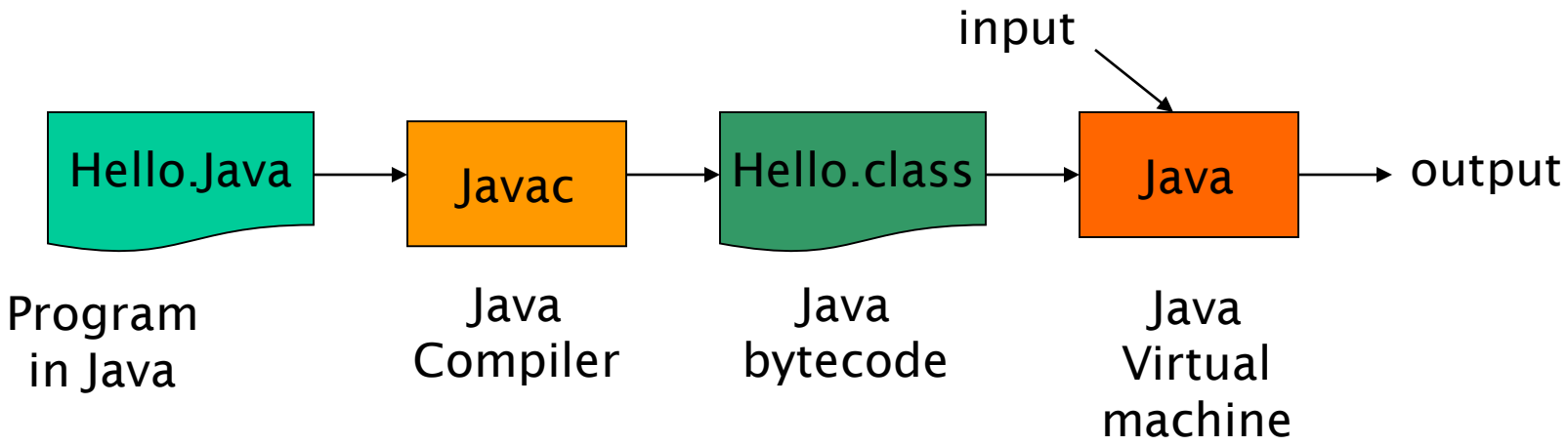
The Java SE Development Kit (JDK)

1. classes.zip (This's needed by the compiler & interpreter.)
2. src.zip (source files for all classes that make up the Java core API – to learn Java)
3. Java Compiler (javac.exe)
4. Java Interpreters (java.exe jre.exe)
5. Java Applet Viewer (appletviewer.exe)
6. Java Debugger (jdb.exe)
7. Java Documentation Generator (javadoc.exe)

etc.

To compile and run a Java program using JDK

```
C:\> javac Hello.java
    (The compiler creates Hello.class)
C:\> java Hello
    (The interpreter excutes Hello.class.)
```



To compile a Java program on line

There are web sites providing remote compiling services, e.g.

http://www.innovation.ch/java/java_compile.html

Input from the Java Console

- Scanner class provides methods for reading *byte*, *short*, *int*, *long*, *float*, *double*, and *String* data types from the Java console or text files.
- *Scanner* is in the *java.util* package.
- Scanner **parses** (separates) input into sequences of characters called **tokens**.
- By default, tokens are separated by standard white space characters (tab, space, newline, etc).

A *Scanner* Constructor

Scanner(InputStream source)

creates a *Scanner* object for reading from *source*. If *source* is *System.in*, this instantiates a *Scanner* object for reading from the Java console

- Example:

```
Scanner stdin = new Scanner( System.in );
```

Scanner next… Methods

Return type	Method name and argument list
dataType	nextDataType( ) returns the next token in the input stream as a *dataType*. *dataType* can be *byte*, *int*, *short*, *long*, *float*, *double*, or *boolean* *i.e.* stdin.nextInt();
String	next( ) returns the next token in the input stream as a *String* *i.e.* stdin.next();
String	nextLine( ) returns the remainder of the line as a *String*

Example

```
import java.util.Scanner;
public class scannerTest {
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        do {     if (stdin.hasNextInt()) {
                        int n = stdin.nextInt();
                        System.out.println("Integer entered: "+n);
                 } else   if (stdin.hasNext("[a-zA-Z]*")) {
                        String s = stdin.next();
                        System.out.println("String entered: "+s);
                 } else break;
        } while (true);
    }
}
```

Output to the Java Console

- *System.out* is the default standard output device, which is tied to the Java Console.
- 3 commonly used methods:
  - `System.out.print("Hello!");`
  - `System.out.println("Hello!");`
  - `System.out.printf("%10s","Hello");`
    - Displays formatted data.

printf()

printf (String format, Object... args)

- The "..." indicates the *varargs* functionality (also referred to as *variable arity methods).*
- Convert values arg1, arg2, etc. to characters according to conversion specifiers.
- A conversion specifier begins with the character % and ends with a conversion character.

Example:

System.out.printf("x = %d and y = %f\n", x, y);

printf() example

```
int x = 12;
float y = 3.5;

System.out.printf("x = %d and y = %f\n", x, y);
```

Conversion specifiers

Output:

```
x = 12 and y = 3.5
```

Conversion Characters

Char	value type	Converted to
b, B	boolean	“True”/”false”
c, C	character	Single character
s, S	String	Character strings
d	int	Signed decimal integer
o	int	Unsigned octal integer
x,X	int	Unsigned hexadecimal
f	double	d.dddddd
e,E	double	d.ddddd$\underline{E+}$xx
g,G	double	In f or e style whichever is smaller
h, H	general	Hex string

Field Width and Precision

1. Put between % **and** conversion character
2. Field width specifies minimum <u>number of columns</u> to be used.
3. If the field width is positive, right justification is used. If the field width is negative, left justification is used.
4. **Period** to separate field width from precision
5. Precision specifies
   - String's maximum number of characters to be printed.
   - Number of digits after the decimal point

Flags

Put between % and the conversion character

- Plus sign (+), to display plus or negative sign for numbers.
  - **System.out.printf("%+d\n", 123)**
- Space, put a space before positive number.
- Pound sign (#), to put 0 for octal, 0x for hexadecimal, force a decimal point for floats.
  - `System.out.printf("%x, %#x\n", 123, 123);`
- Zero (0), to pad a field with leading zeros
  `System.out.printf("%09d\n", 123);`

Example

```
public class printfTest1 {
   public static void main(String[] args) {
       System.out.printf("%9s%9d%9c%9f\n","aloha",5,'Z',5.67);
       System.out.printf("%-9s%-9d%-9c%9f\n","aloha",5,'Z',5.67);
       System.out.printf("%-9.3s%-9.3f\n","aloha",5.67);
       System.out.printf("%x, %#x\n", 123, 123);
       System.out.printf("%09d\n", 123);
   }
}
```

```
Output:
    aloha        5        Z 5.670000
aloha    5        Z        5.670000
alo      5.670
7b, 0x7b
000000123
```

Argument index

Format: n$

Example:

```
public class printfTest {
   public static void main(String[] args) {
   System.out.printf("%1$d %1$d %3$d %2$d\n",1,2,3,4,5);
   }
}
```

Output:
1 1 3 2

Libraries

- Java programs are usually not written from scratch.
- There are hundreds of library classes for all occasions.
- Library classes are organized into packages. For example:
  - java.util — miscellaneous utility classes
  - java.awt — window and graphics toolkit
  - javax.swing — GUI development package

Import library classes

- A fully-qualified library class name includes the package name. For example:
  - **java.awt.**Color

    package class
  - **javax.swing.**JButton
- Import statements at the top of the source file let you refer to library classes by their short names.

```
import   javax.swing.JButton;
JButton go = new JButton("Go");
```

Fully-qualified name

short name

Import library classes conti.

- You can import all classes of a package by using a wildcard .*:
  - import java.awt.*;
  - import java.awt.event.*;
  - import javax.swing.*;

Imports all classes from **awt**, **awt.event**, and **swing** packages

- java.lang is imported automatically into all classes; defines System, Math, Object, String, and other commonly used classes.

Homework.

Show lines printed.

```
System.out.printf("%d\n", 123);
System.out.printf("%+d\n", 123);
System.out.printf("%10d\n", 123);
System.out.printf("%010d\n", 123);
System.out.printf("%-10d\n", 123);
System.out.printf("%x\n", 123);
```

column	0	1	2	3	4	5	6	7	8	9	10	11
Line 1												
Line 2												
Line 3												
Line 4												
Line 5												
Line 6												