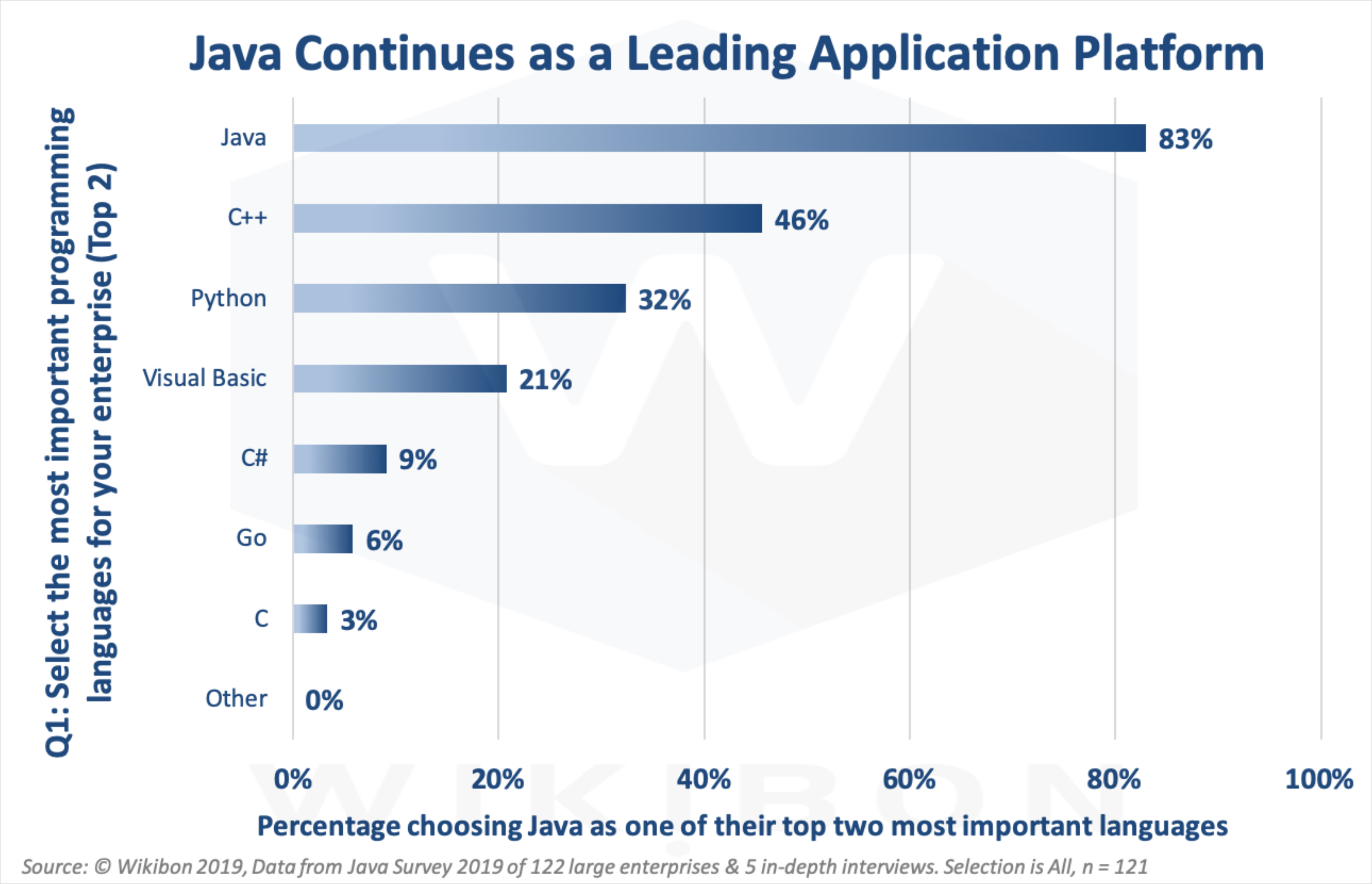
Java Basics - Part 1

What is Java

- Java is the #1 programming language and development platform.



Source: © Wikibon 2019, Data from Java Survey 2019 of 122 large enterprises & 5 in-depth interviews. Selection is All, n = 121

What is Java

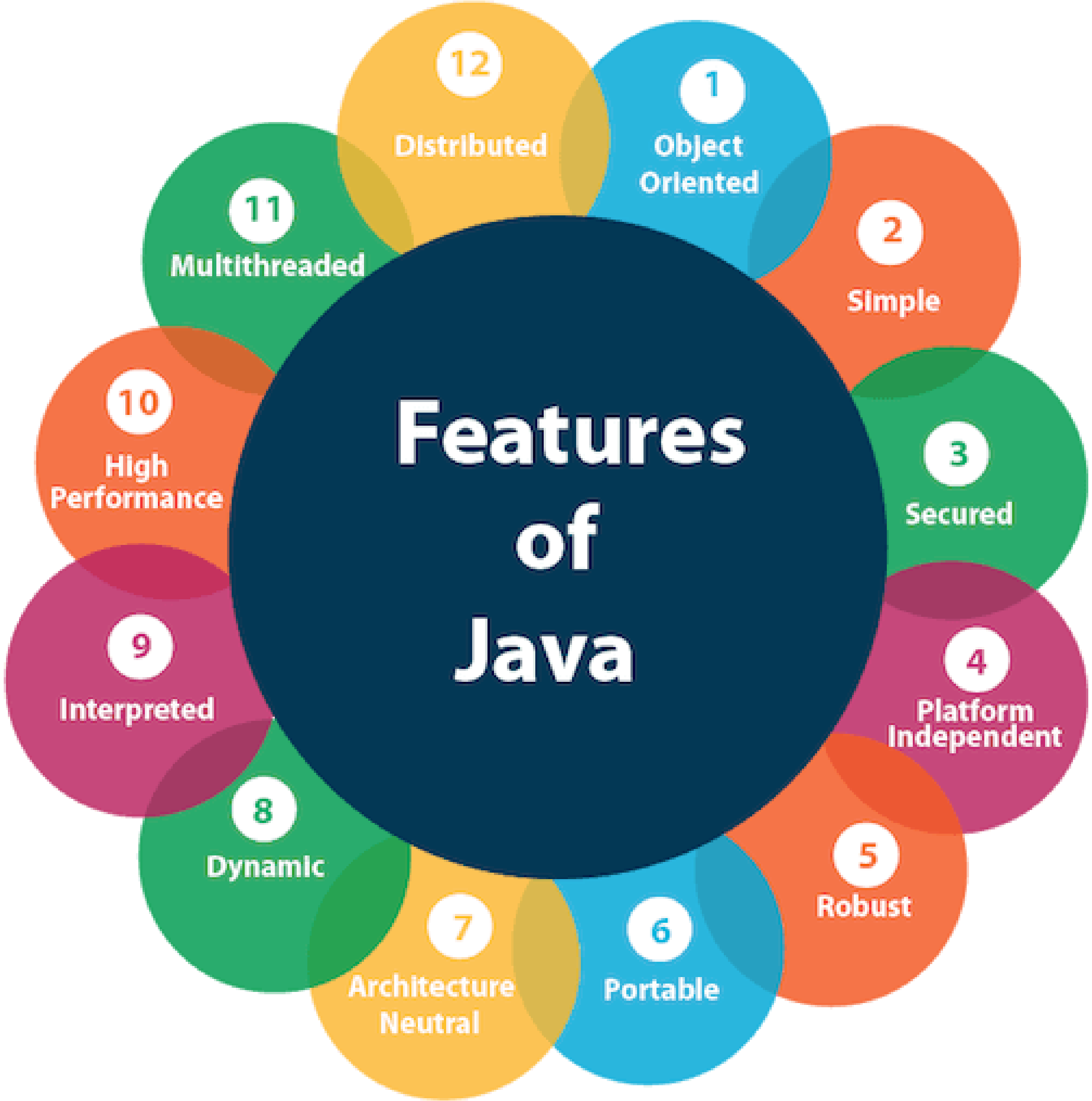
- According to its developer.
  - ► With millions of developers running more than 51 billion Java Virtual Machines worldwide, Java continues to be the development platform of choice for enterprises and developers.”
    - from https://www.oracle.com/java/

```
var now = new Date();
var hours = now.getHours();
var minutes = now.getMinutes();
var seconds = now.getSeconds();

var ampm = "am";
var colon = '<IMG SRC="images/colon.gif">';

if (hours >= 12) {
     ampm = "pm";
     hours = hours - 12;
}

if (hours == 0) hours = 12;

if (hours < 10) hours = "0" + hours;
else hours = hours + '';

if (minutes < 10) minutes = "0" + minutes;
else minutes = minutes + '';

if (seconds < 10) seconds = "0" + seconds;
else seconds = seconds + '';
```

Why Java?

- Java so popular for developers and programmers
  - Watch this: https://www.youtube.com/watch?v=Zv8-hrGiGno
  - And read this:

Why Java


Features of Java
1 Object Oriented
2 Simple
3 Secured
4 Platform Independent
5 Robust
6 Portable
7 Architecture Neutral
8 Dynamic
9 Interpreted
10 High Performance
11 Multithreaded
12 Distributed

History of Java

- Who developed it?
- For what purpose?
- Why was it called Java?
- Why is it signified/pictured with a coffee cup?

Sun microsystems

ORACLE

Certified Associate

Java SE 8 Programmer

Java

History of Java

- Java, whose original name was Oak, was developed by Sun Microsystems in 1991, created for consumer electronics (embedded software).

- Internet and Web was just emerging, so Sun turned it into a language of Internet programming.

- Sun was acquired by Oracle in 2010, for $7.4 billion.

Sun
microsystems

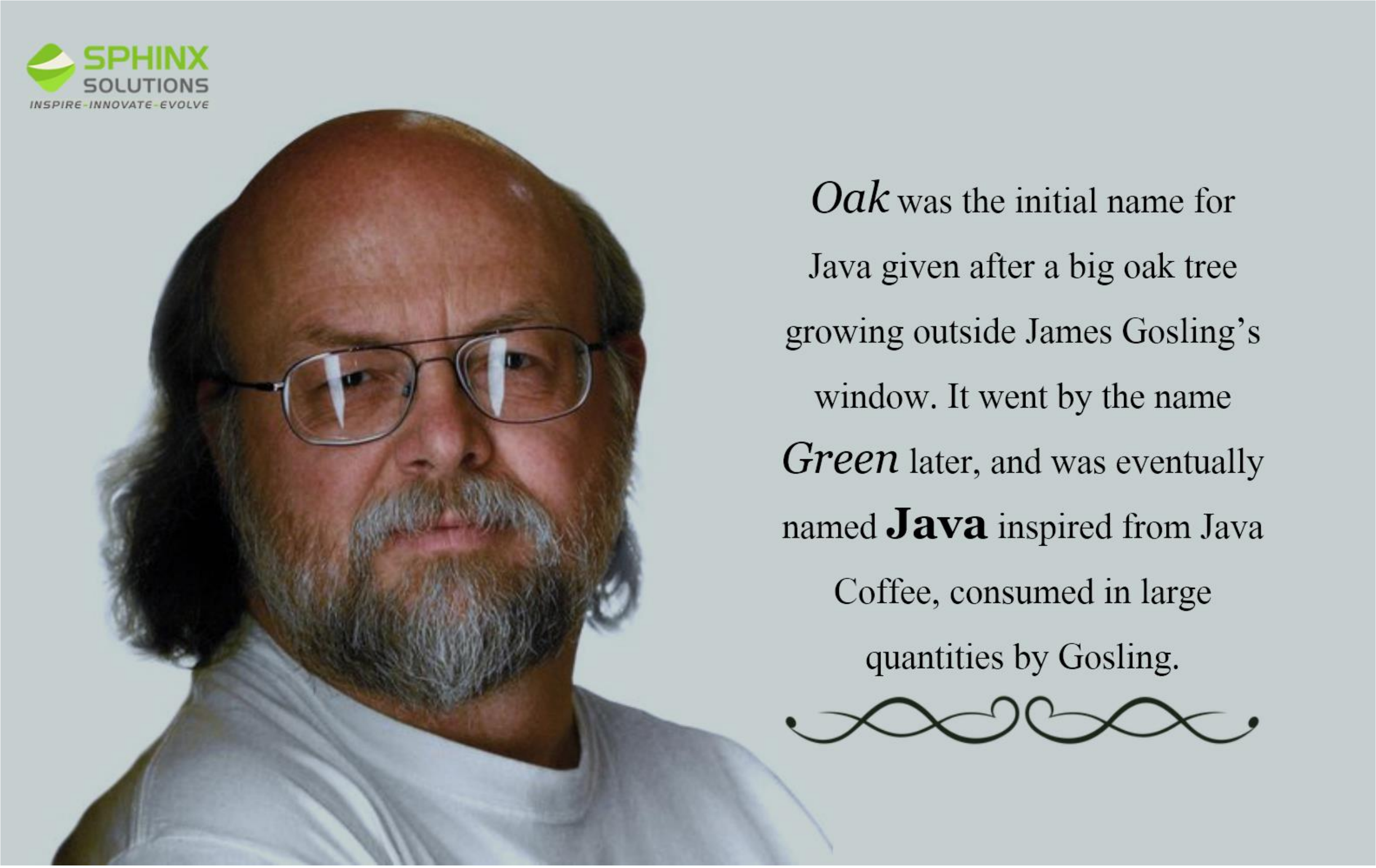
ORACLE
Certified Associate
Java SE 8 Programmer

History of Java

- Java project was started by a Green team in Sun, with key members including James Gosling
  - ► trying to figure out what would be the "next wave" of computing and how we might catch it
  - ► at least one of the waves was going to be the convergence of digitally controlled consumer devices and computers
  - ► James known for ‘the father of Java’

History of Java

- How about the names - ‘oak’, ‘Java’?
  - More here: https://www.javatpoint.com/history-of-java

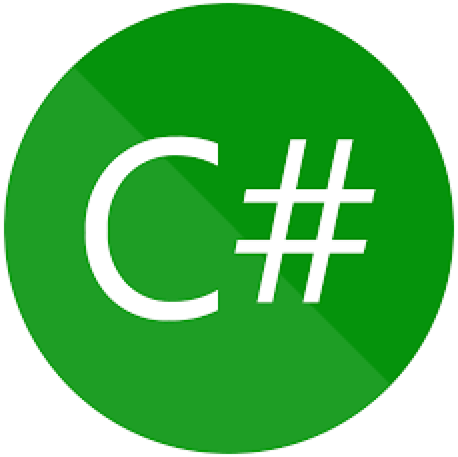


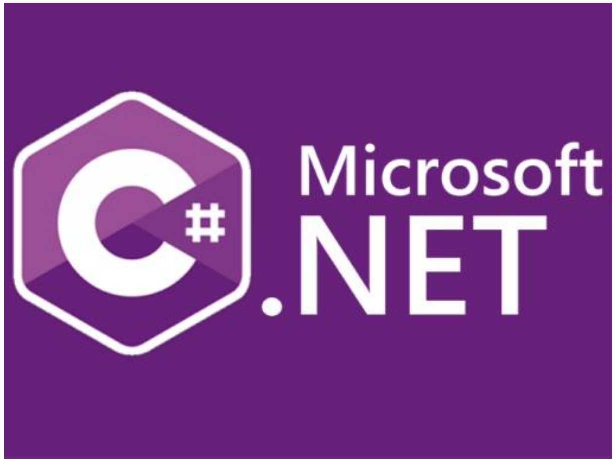
Oak was the initial name for Java given after a big oak tree growing outside James Gosling’s window. It went by the name *Green* later, and was eventually named **Java** inspired from Java Coffee, consumed in large quantities by Gosling.

Similar to Java …

- Another programming language takes the same approach …

- That is C#, pronunciated as C Sharp



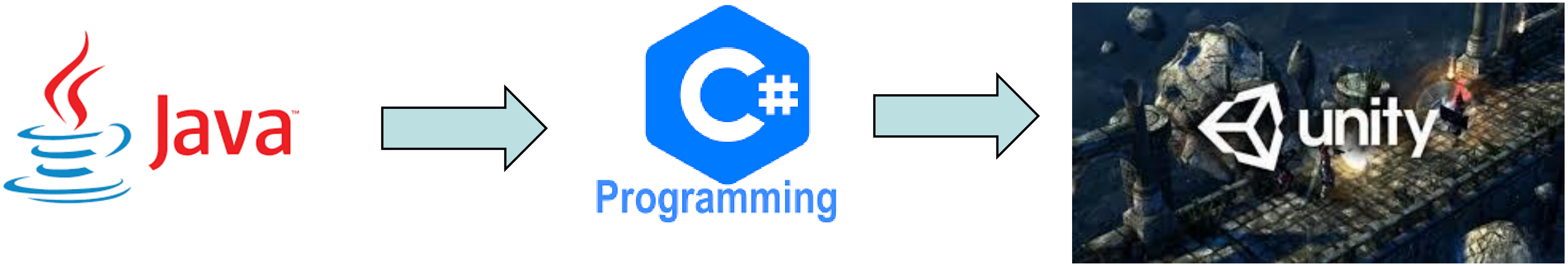


Similar to Java …

- Originated by Microsoft in 2000, as a response to Java, enabling programmers to migrate from C/C++ and Java easily
- Why is it called ‘C Sharp’?
- Initially called ‘Cool’, and later inspired by music note ‘C#’ – a step above ‘C’

Similar to Java …

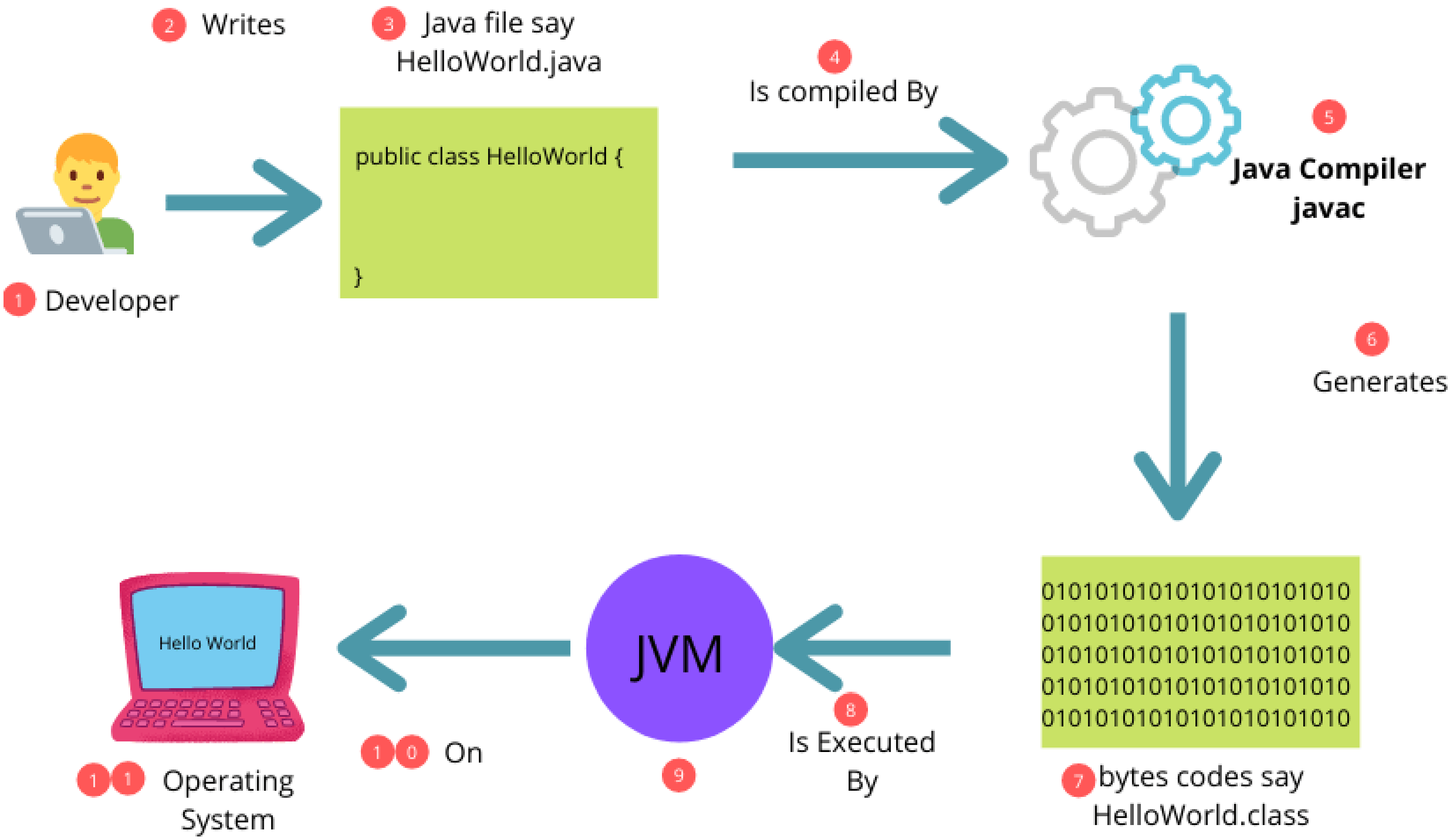
- Since Java and C# are similar, you shall be able to code in C# comfortably after you learn Java
  - Another advantage of learning Java, isn't it?
  - C# for Unity Engine
    - – Develop your first game with Unity, Wow!



Java vs Python

- Which one is more popular? Which one shall I learn if I have to choose?

How Java Works



1 Developer

2 Writes

3 Java file say
HelloWorld.java

```
public class HelloWorld {

}
```

4 Is compiled By

5 Java Compiler
javac

6 Generates

```
01010101010101010101010
01010101010101010101010
01010101010101010101010
01010101010101010101010
01010101010101010101010
```

7 bytes codes say
HelloWorld.class

8 Is Executed
By

9 JVM

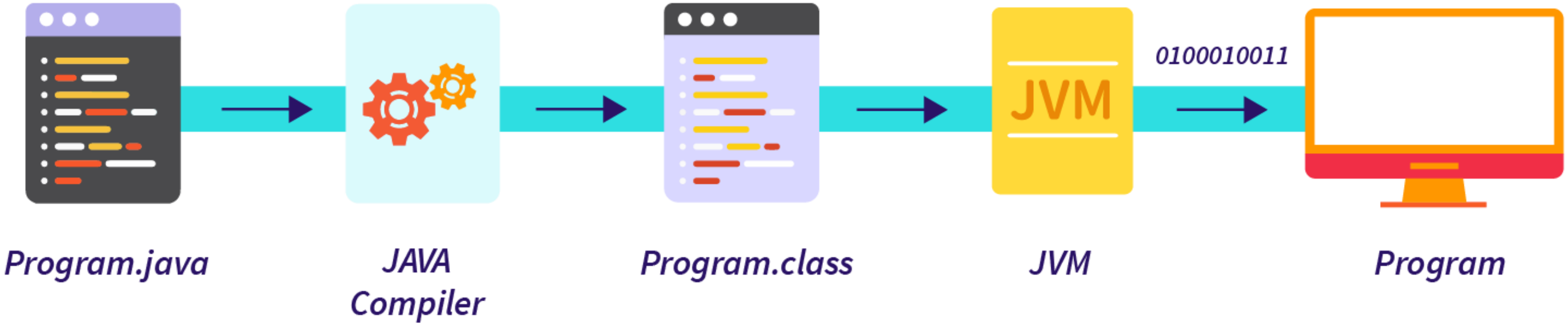
10 On

11 Operating
System

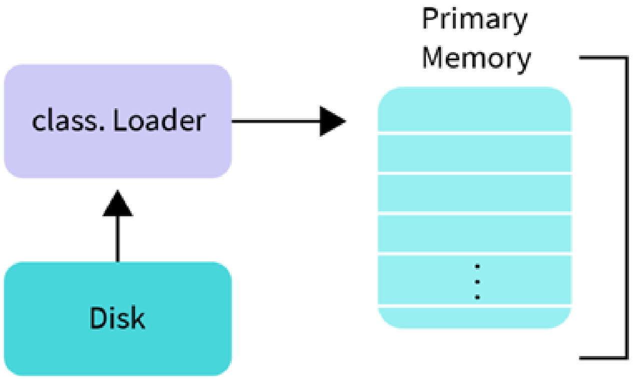
https://www.google.com/url?sa=i&url=https%3A%2F%2Ftutorialshub.org%2Fhow-java-works%2F&psig=AOvVaw2iY26oDDezsD_c1paq8oH6&ust=1666749512006000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCMjD4faj-voCFQAAAAAdAAAAABAT

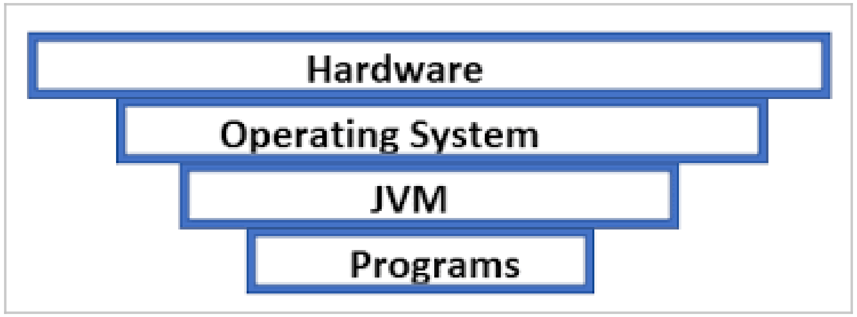
How Java Works

- Overall procedure



- Java Virtual Machine (JVM) – platform dependent
  - Read .class files from disk





- Now let’s have a look at the most famous example of program, nearly for all programming language!

- What’s that? Have a guess?

HelloWorld.java

```java
/**
 * A simple program
 */

public class HelloWorld
{
        public static void main(String[] args)
        {
                System.out.println("HELLO CompSci 201!");
        }
}
```

HelloWorld.java

```
public class HelloWorld
{
        public static void main(String[] args)
        {
                System.out.println("HELLO CompSci 201!");
        }
}
```

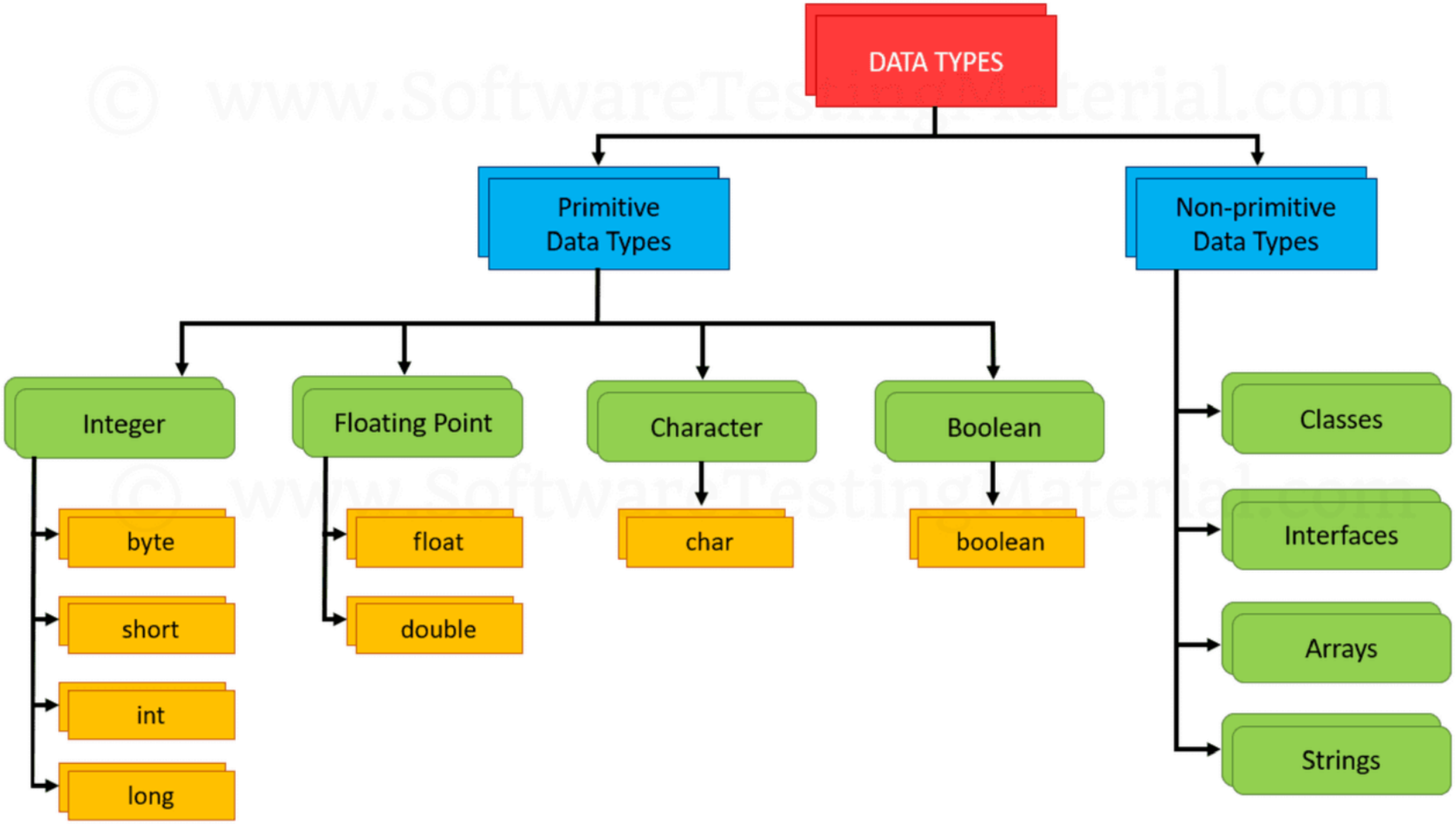
- The code for class HelloWorld will be in a file named HelloWorld.java
  - just a text file with the .java extension
  - a class is a programmer defined data type

- A complete program/project will normally consist of many different classes and thus many different files

Basic Features

- Data Types
  - primitives
  - Non-primitives
- Expressions and operators
- Arrays
- Loops and controls

Data Types



https://1.bp.blogspot.com/-XiCil0D2LnM/YOqYdFwT3jI/AAAAAAAAoiI/_LXknQ6AS78--pIjDY9nREG_hsbUyOu6QCLcBGAsYHQ/s1297/Data%2Btypes%2Bin%2BJava.png

Primitive Data Types

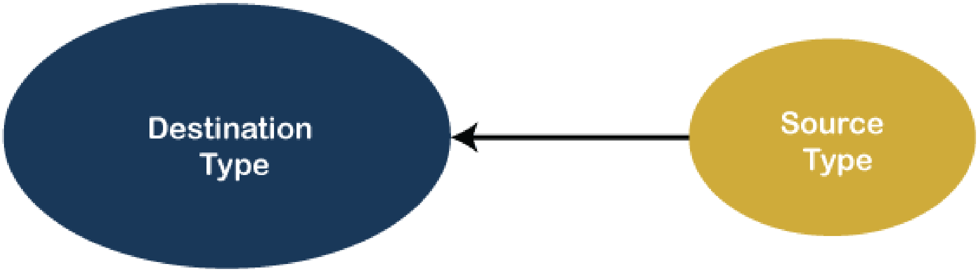
- Integers: byte, short, int and long
- Floating point: float, double
- Boolean: true, false
- Character: char

Primitive Data Types - bits

Data Type	Characteristics	Range
`byte`	8 bit signed integer	-128 to 127
`short`	16 bit signed integer	-32768 to 32767
`int`	32 bit signed integer	-2,147,483,648 to 2,147,483,647
`long`	64 bit signed integer	-9,223,372,036,854,775,808 to- 9,223,372,036,854,775,807
`float`	32 bit floating point number	± 1.4E-45 to ± 3.4028235E+38
`double`	64 bit floating point number	± 4.9E-324 to ± 1.7976931348623157E+308
`boolean`	`true` or `false`	NA, note Java booleans cannot be converted to or from other types
`char`	16 bit, Unicode	Unicode character, `\u0000` to `\uFFFF` Can mix with integer types

Type conversion

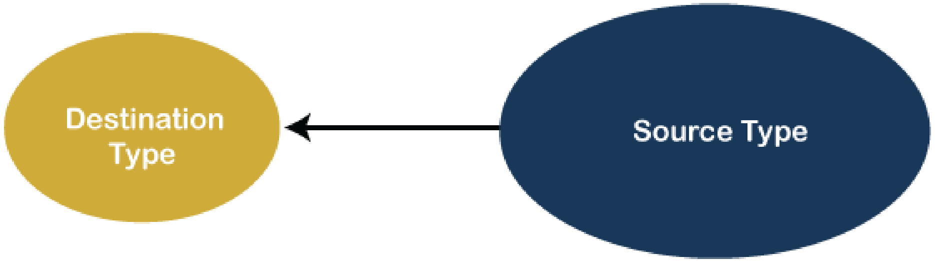
- a data type is automatically converted into another data type at compile time
  - the destination data type should not be smaller than the source type. Also known as widening conversion of the data type.



```
int x = 3, y = 4;  // the size of int variable is 4
byte.
double area; // the size of float variable is 8 bytes.
area =  x * y; /* automatic conversion by the compiler
at the compile time of a program. */
```

Type casting

- a data type is converted into another data type by a programmer
  - the destination data type must be smaller than the source data type. Hence it is also called a narrowing conversion.



```
double x = 3.5, y = 4.5;   // the size of double
variable is 4 byte.
int area; // the size of the int variable is 2 bytes.
area = (int) x * y; // after conversion the product
converts into integer
```

Type conversion / casting

- Will the following work?

```
int iNum = 10;
long lNum = iNum;
double dNum = lNum;
```

```
double dNum = 100.95;
long lNum = (long)dNum;
int iNum = (int)lNum;
System.out.println("Double value: " + dNum);
System.out.println("long value: " + lNum);
```

Type conversion / casting

- Will the following work?

```
// Declaring character variable
char ch = 'c’;

// Declaring integer variable
int num = 88;

// inserting integer to character
ch = num;
```

```
int i = 10;
float f = 9.9;
String s = i>d? "yes": "no";
System.out.println("i is bigger than d? " + s);
```

Operators

- arithmetic : +, -, *, /, % (modulo, reminder)
- comparison: <, <=, >, >=, ==, !=
- pre/post-increment: ++ --
- logical: and &&, or ||, not !
- conditional: cond ? true-exp : false-exp
- assignment: =
- compound assignment: +=, -=, *=, /=

String

- String is a very useful data type

```
String myName = “Jianlin Wang”;
String courseTitle = “Intro to Programming”;

System.out.println(courseTitle);
```

- Concatenation:

```
String s1 = “The title of the course is “;
String s2 = s1 + courseTitle;

System.out.prinln(s2);
```

Input/output

- Simple input and output in Java are realized through
  - System.out, that performs output to the “standard output” device

```
// Print the string s
print(String s);

// Similar to print(s),
// followed by the newline character
println(String s);
```

Input/output

- Simple input and output in Java are realized through
  - System.in, for performing input from the Java console window
    - Use the class ‘Scanner’ in the ‘utility’ package to take user’s input

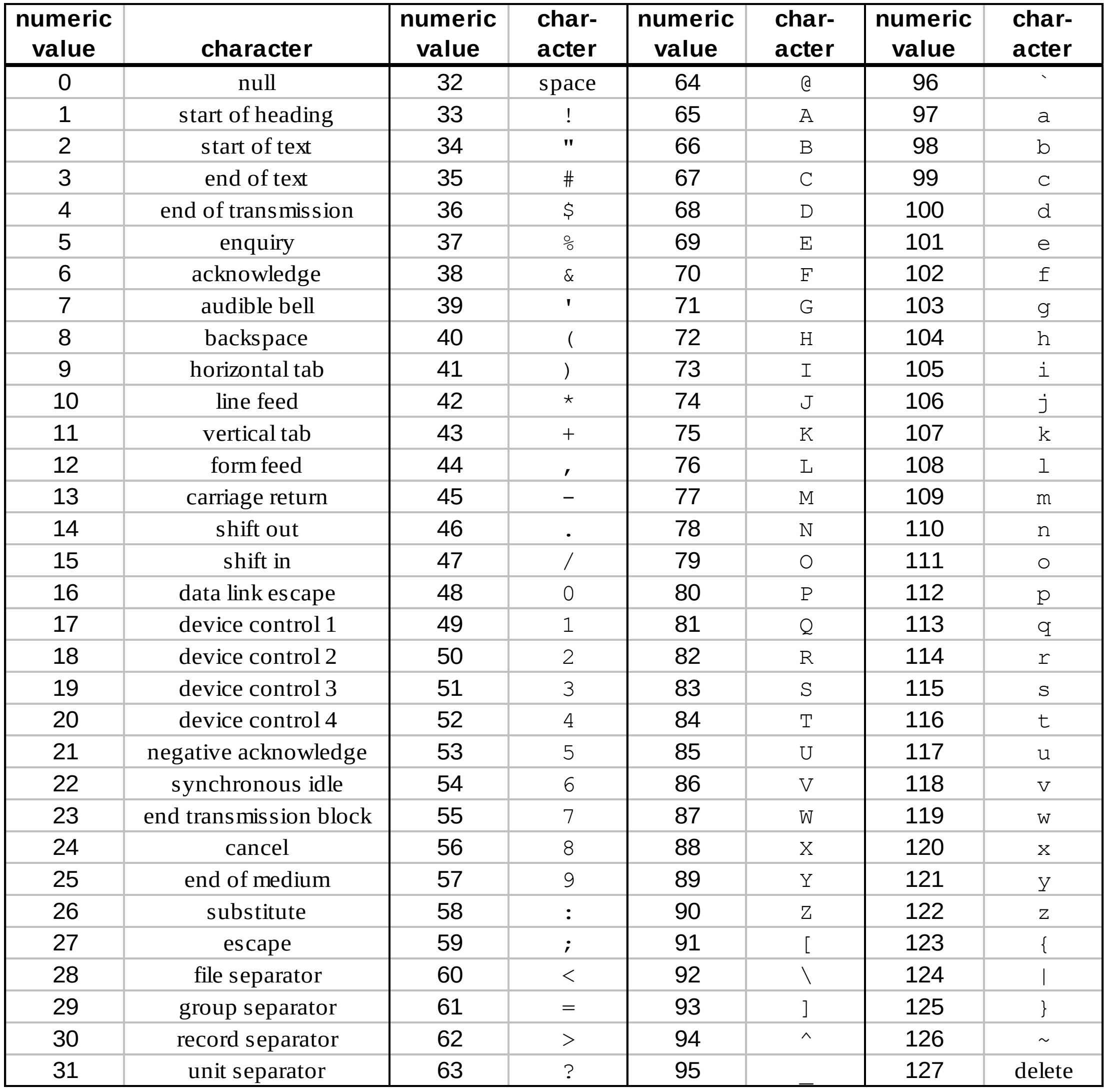
```
 Scanner input = new Scanner(System.in);

 System.out.print("Enter your age: ");
 double age = input.nextDouble( );

 System.out.print("Enter your maximum heart rate: ");
 double rate = input.nextDouble( );

 double fb = (rate-age) * 0.65;
 System.out.println("Your ideal fat-burning heart rate
is " + fb);
```

Input/output

- Example question:
  - ► Write a program that asks user to enter two integers, x and y, and prints the Euclidean distance from the point (x, y) to the origin (0, 0).
    - Euclidean is distance is defined as sqrt(x^2 + y^2).
  - ► Try yourself
    - Or work with someone beside you, especially when your Eclipse does not work yet

Input/output

- ‘char’ and ASCII
  - character values have an underlying numeric value. For example, the letter 'A' has the underlying value of 65.
  - ASCII: American Standard Code for Information Interchange
- Why underlying numeric value?
  - So characters can be ordered (e.g., 'A' comes before 'B' because A's 65 is less than B's 66).



numeric value	character	numeric value	char-acter	numeric value	char-acter	numeric value	char-acter
0	null	32	space	64	@	96	`
1	start of heading	33	!	65	A	97	a
2	start of text	34	"	66	B	98	b
3	end of text	35	#	67	C	99	c
4	end of transmission	36	$	68	D	100	d
5	enquiry	37	%	69	E	101	e
6	acknowledge	38	&	70	F	102	f
7	audible bell	39	'	71	G	103	g
8	backspace	40	(	72	H	104	h
9	horizontal tab	41	)	73	I	105	i
10	line feed	42	*	74	J	106	j
11	vertical tab	43	+	75	K	107	k
12	form feed	44	,	76	L	108	l
13	carriage return	45	-	77	M	109	m
14	shift out	46	.	78	N	110	n
15	shift in	47	/	79	O	111	o
16	data link escape	48	0	80	P	112	p
17	device control 1	49	1	81	Q	113	q
18	device control 2	50	2	82	R	114	r
19	device control 3	51	3	83	S	115	s
20	device control 4	52	4	84	T	116	t
21	negative acknowledge	53	5	85	U	117	u
22	synchronous idle	54	6	86	V	118	v
23	end transmission block	55	7	87	W	119	w
24	cancel	56	8	88	X	120	x
25	end of medium	57	9	89	Y	121	y
26	substitute	58	:	90	Z	122	z
27	escape	59	;	91	[	123	{
28	file separator	60	<	92	\	124	|
29	group separator	61	=	93	]	125	}
30	record separator	62	>	94	^	126	~
31	unit separator	63	?	95	_	127	delete

Input/output

- Print out ‘char’ – what’s the output of the following?

```
char first = ‘A’;
System.out.println(first);
```

```
char first = ‘A’;
int asci = first;
System.out.println(“The asci code of ‘A’
is “ + asci);
```

```
char first = ‘A’;
char second = ‘B’;
System.out.println(first + second);
```

```
char first = ‘A’;
char second = ‘B’;
System.out.println(“The concatenated
characters are “ + first + second);
```

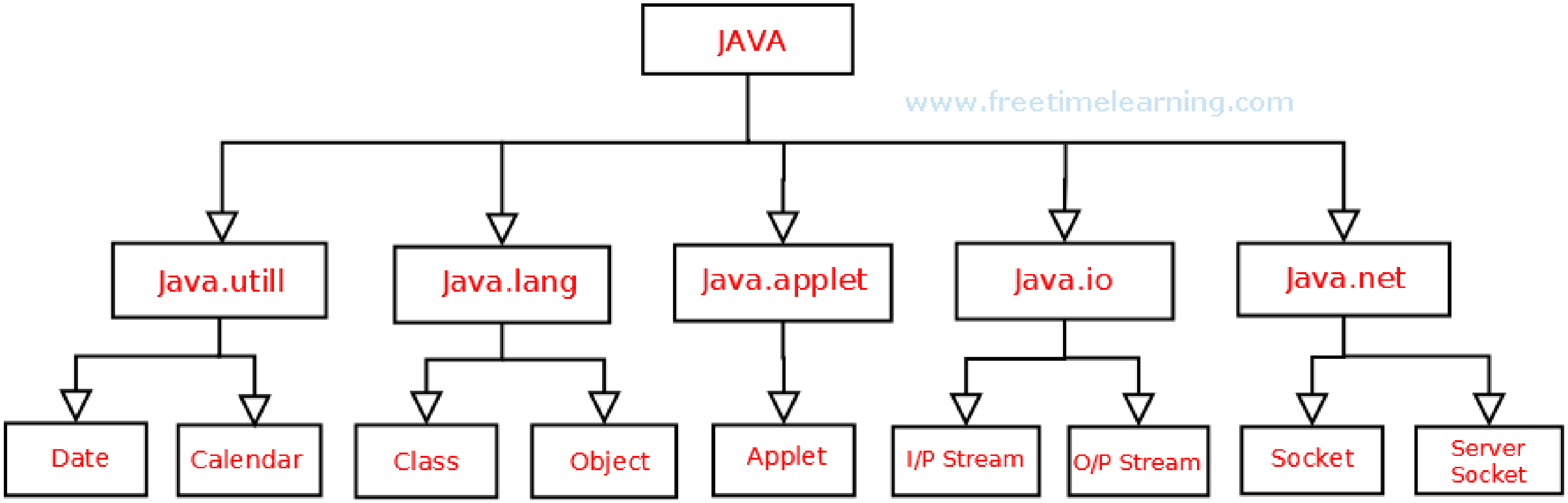
Packages and import

- Java language takes a general and useful approach to the organization of classes into programs.

- To aid in the organization of large code repository, Java allows a group of related classes to be grouped into what is known as a package.

Packages and import

- Java has many built-in packages that consist of many classes (and methods) for programmer to use.



Packages and import

- You may import them by
  - Individual class from a package

```
import java.util.Scanner;

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

  - Whole package

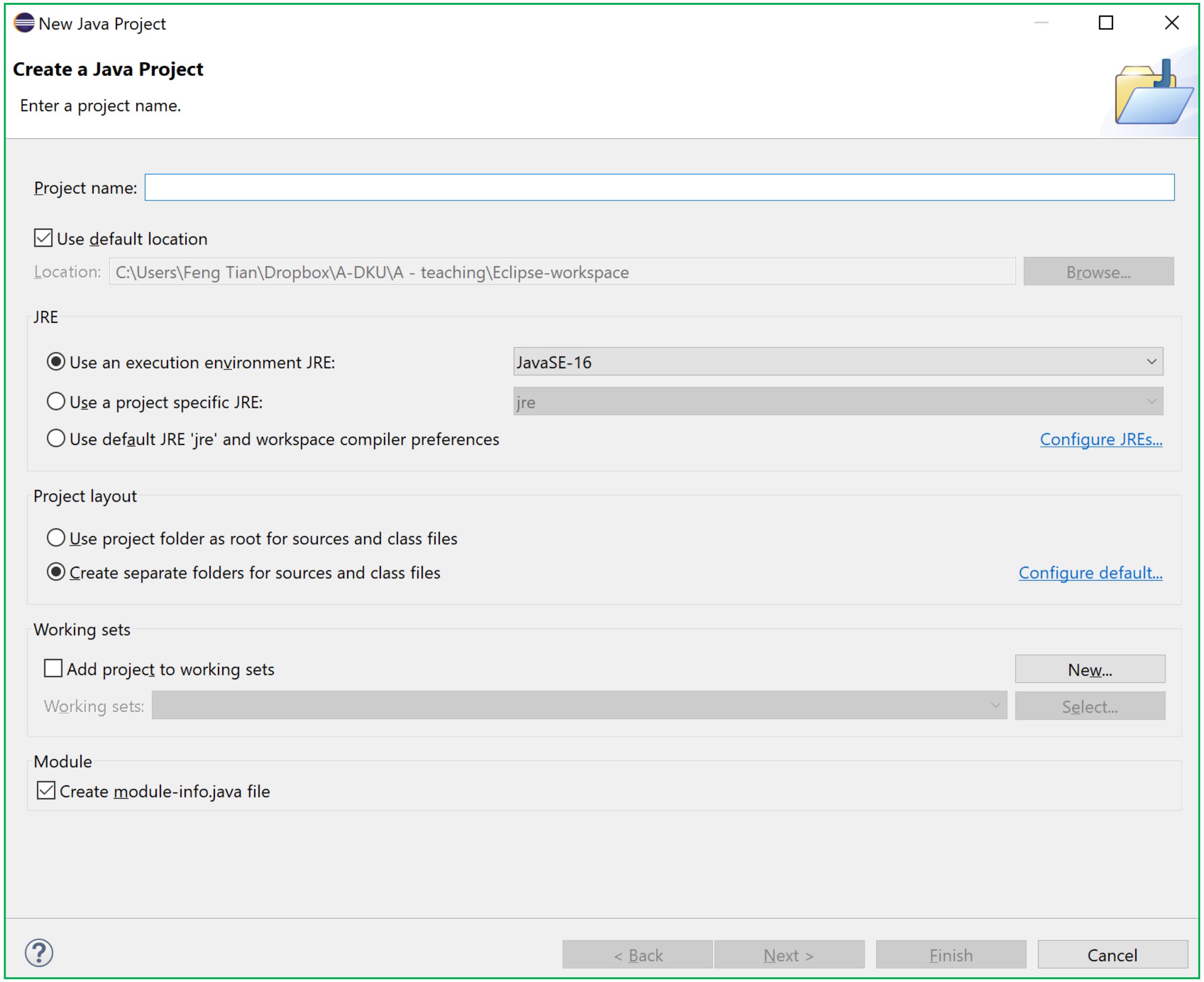
```
import java.util.*;

Import java.lang.*;
```

Packages and import

- Advantages of using package:
  - Help us avoid the pitfalls of name conflicts. If all type definitions were in a single package, there could be only one public class named Window. But with packages, we can have an *architecture.Window* class that is independent from a *gui.Window* class for graphical user interfaces.
  - It is much easier to distribute a comprehensive set of classes for other programmers to use when those classes are packaged.
  - When type definitions have a related purpose, it is often easier for other programmers to find them in a large library and to better understand their coordinated use when they are grouped as a package.

Modules

- Modularity of Java
  - Starting from Java SE9, Java has introduced a new concept or mechanism, called ‘module’
  - When you create a Java project in Eclipse, a default option is to create a module file



New Java Project

Create a Java Project

Enter a project name.

Project name:

☑ Use default location

Location: C:\Users\Feng Tian\Dropbox\A-DKU\A - teaching\Eclipse-workspace

JRE

● Use an execution environment JRE: JavaSE-16

○ Use a project specific JRE: jre

○ Use default JRE 'jre' and workspace compiler preferences

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

Working sets

□ Add project to working sets

Working sets:

Module

☑ Create module-info.java file

Modules

- Modularity of Java
  - A higher level above the package
  - More information: https://www.oracle.com/corporate/features/understanding-java-9-modules.html

- About homework
  - To be released in Sakai
  - You submit the solutions in Sakai by deadline
  - See an example here