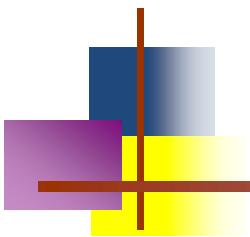




50311



JAVA TECHNOLOGY

CHAPTER 1: INTRODUCTION TO JAVA DEVELOPMENT



History of Java

- Java was originally developed by Sun Microsystems starting in 1991
 - James Gosling
 - Patrick Naughton
 - Chris Warth
 - Ed Frank
 - Mike Sheridan
- This language was initially called *Oak*
- Renamed **Java** in 1995

What is Java

- A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language -- **Sun Microsystems**
- **Object-Oriented**
 - No free functions
 - All code belong to some class
 - Classes are in turn arranged in a hierarchy or package structure

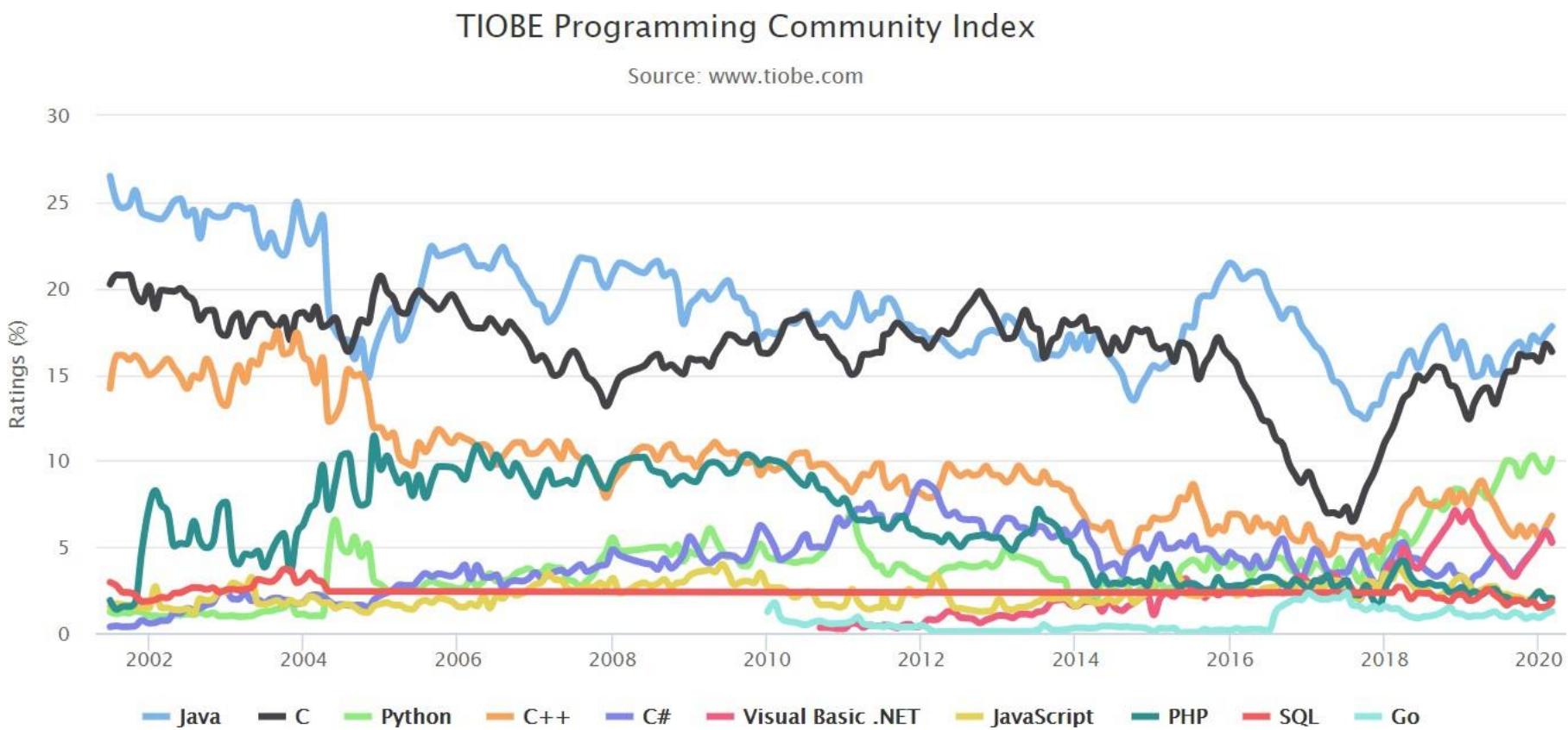
What is Java

- **Distributed**
 - Fully supports IPv4, with structures to support IPv6
 - Includes support for Applets: small programs embedded in HTML documents
- **Interpreted**
 - The programs are compiled into Java Virtual Machine (JVM) code called bytecode
 - Each bytecode instruction is translated into machine code at the time of execution

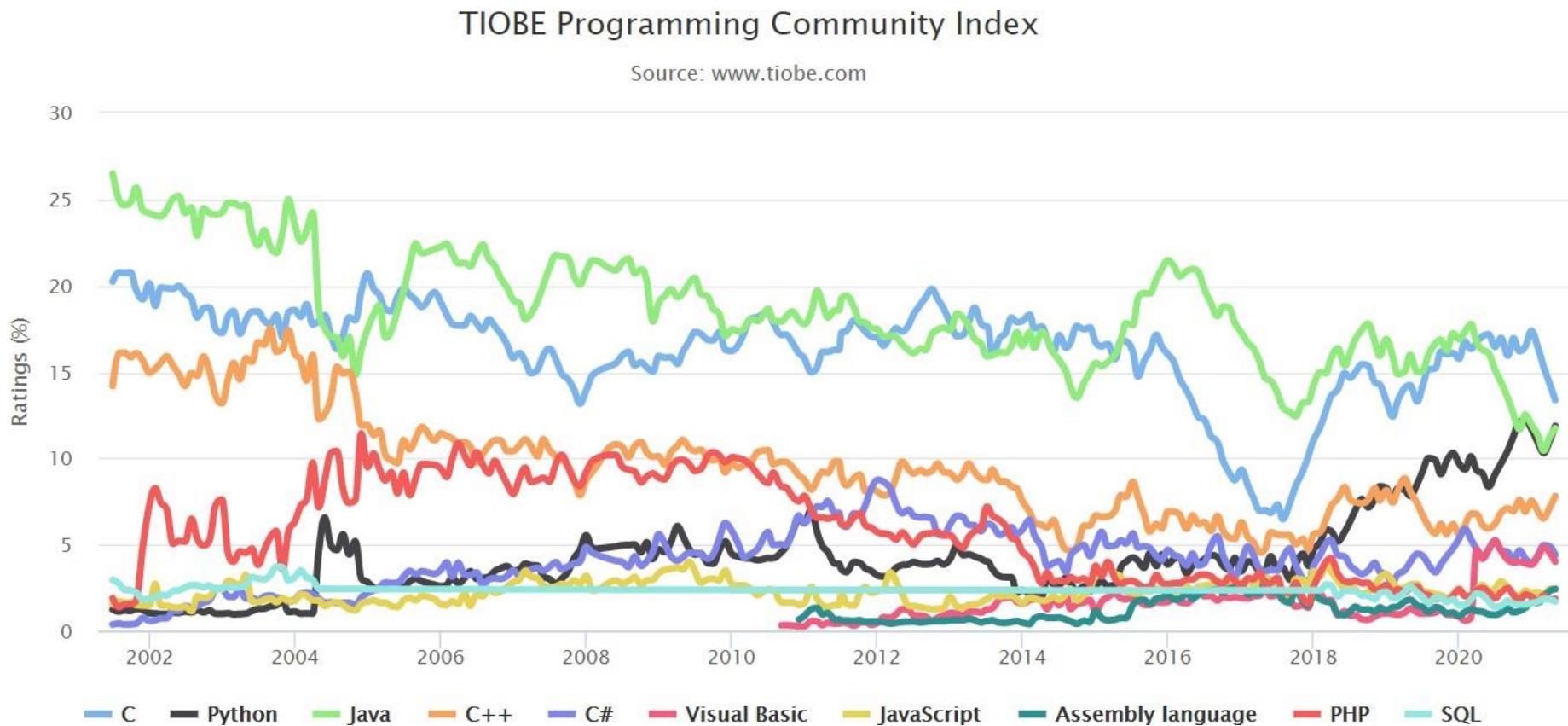
What is Java

- **Robust**
 - Java is simple – no pointers/stack concerns
 - Exception handling – try/catch/finally series allows for simplified error recovery
 - Strongly typed language – many errors caught during compilation

Java – The Most Popular (2020)



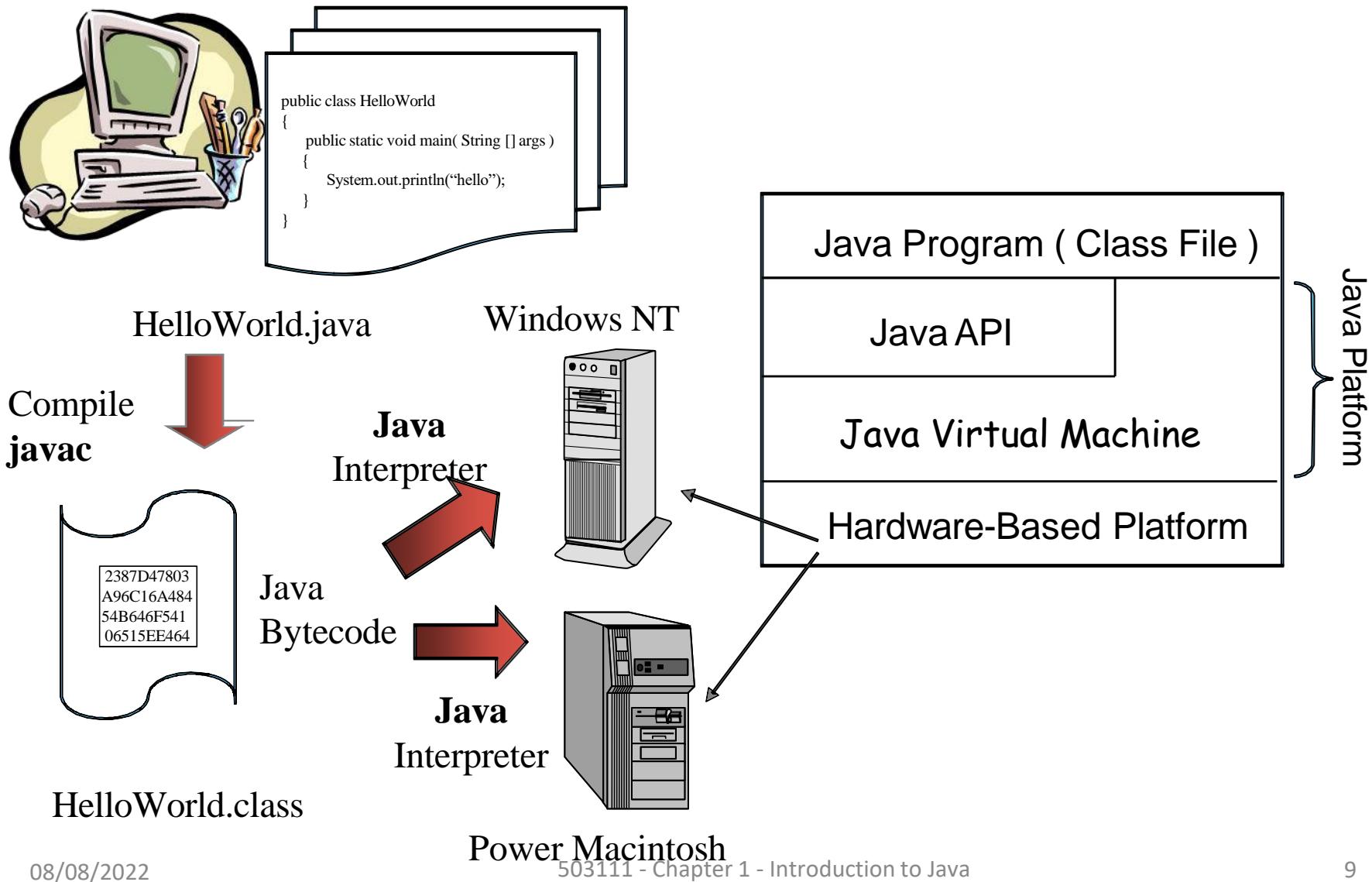
Java – Top Three (2021)



Java Editions

- Java 2 Platform, Standard Edition (J2SE)
 - Used for developing Desktop based application and networking applications
- Java 2 Platform, Enterprise Edition (J2EE)
 - Used for developing large-scale, distributed networking applications and Web-based applications
- Java 2 Platform, Micro Edition (J2ME)
 - Used for developing applications for small memory-constrained devices, such as cell phones, pagers and PDAs

Java platform



Java Development Environment

- Edit
 - Create/edit the source code
- Compile
 - Compile the source code
- Load
 - Load the compiled code
- Verify
 - Check against security restrictions
- Execute
 - Execute the compiled

Phase 1: Creating a Program

- Any text editor or Java IDE (Integrated Development Environment) can be used to develop Java programs
- Java source-code file names must end with the *.java* extension
- Some popular Java IDEs are
 - NetBeans
 - Eclipse
 - IntelliJ

Phase 2: Compiling a Java Program

- *javac Welcome.java*
 - Searches the file in the current directory
 - Compiles the source file
 - Transforms the Java source code into bytecodes
 - Places the bytecodes in a file named **Welcome.class**

Bytecodes *

- They are not machine language binary code
- They are independent of any particular microprocessor or hardware platform
- They are platform-independent instructions
- Another entity (interpreter) is required to convert the bytecodes into machine codes that the underlying microprocessor understands
- This is the job of the **JVM** (Java Virtual Machine)

JVM (Java Virtual Machine) *

- It is a part of the JDK and the foundation of the Java platform
- It can be installed separately or with JDK
- A virtual machine (VM) is a software application that simulates a computer, but hides the underlying operating system and hardware from the programs that interact with the VM
- It is the JVM that makes Java a portable language

JVM (Java Virtual Machine) *

- The same bytecodes can be executed on any platform containing a compatible JVM
- The JVM is invoked by the java command
 - *java Welcome*
- It searches the class Welcome in the current directory and executes the main method of class Welcome
- It issues an error if it cannot find the class Welcome or if class Welcome does not contain a method called main with proper signature

Phase 3: Loading a Program *

- One of the components of the JVM is the class loader
- The class loader takes the .class files containing the programs bytecodes and transfers them to RAM
- The class loader also loads any of the .class files provided by Java that our program uses

Phase 4: Bytecode Verification *

- Another component of the JVM is the bytecode verifier
- Its job is to ensure that bytecodes are valid and do not violate Java's security restrictions
- This feature helps to prevent Java programs arriving over the network from damaging our system

Phase 5: Execution

- Now the actual execution of the program begins
- Bytecodes are converted to machine language suitable for the underlying OS and hardware
- Java programs actually go through two compilation phases
 - Source code -> Bytecodes
 - Bytecodes -> Machine language

Editing a Java Program

The screenshot shows a Java code editor with the file "Welcome.java" open. The code defines a class "Welcome" with a main method that prints two messages to the console: "Hello Java" and "I like %s\n, Java". It also declares a string variable "strDepartment" and prints a message including it. The code ends with a note: "NOTE: no semicolon is required here". The code editor has syntax highlighting and line numbers.

```
1
2 ► public class Welcome {
3 ►   public static void main(String[] args) {
4     System.out.println("Hello Java");
5     System.out.printf("I like %s\n", "Java");
6     String strDepartment = "CSE";
7     System.out.print("We study in " + strDepartment + "\n");
8   } // end method main
9 } // end class Welcome - NOTE: no semicolon is required here
10
```

Examining Welcome.java

- A Java source file can contain multiple classes, but only one class can be a public class
- Typically Java classes are grouped into packages (similar to namespaces in C++)
- A public class is accessible across packages
- The source file name must match the name of the public class defined in the file with the .java extension

Examining Welcome.java

- In Java, there is no provision to declare a class, and then define the member functions outside the class
- Body of every member function of a class (called method in Java) must be written when the method is declared
- Java methods can be written in any order in the source file
- A method defined earlier in the source file can call a method defined later

Examining Welcome.java

- ***public static void main(String[] args)***
 - **main** is the starting point of every Java application
 - **public** is used to make the method accessible by all
 - **static** is used to make main a static method of class Welcome. Static methods can be called without using any object; just using the class name. JVM call main using the **ClassName.methodName** (*Welcome.main*) notation
 - **void** means main does not return anything
 - **String args[]** represents an array of String objects that holds the command line arguments passed to the application. *Where is the length of args array?*

Examining Welcome.java

- Think of JVM as a outside Java entity who tries to access the main method of class Welcome
 - main must be declared as a public member of class Welcome
- JVM wants to access main without creating an object of class Welcome
 - main must be declared as static
- JVM wants to pass an array of String objects containing the command line arguments
 - main must take an array of String as parameter

Examining Welcome.java

- ***System.out.println()***
 - Used to print a line of text followed by a new line
 - **System** is a class inside the Java API
 - **out** is a public static member of class System
 - **out** is an object of another class of the Java API
 - **out** represents the standard output (similar to stdout or cout)
 - **println** is a public method of the class of which out is an object

Examining Welcome.java

- **System.out.print()** is similar to **System.out.println()**, but does not print a new line automatically
- **System.out.printf()** is used to print formatted output like printf() in C
- In Java, characters enclosed by double quotes ("") represents a String object, where String is a class of the Java API
- We can use the plus operator (+) to concatenate multiple String objects and create a new String object

Compiling a Java Program

- Place the .java file in the bin directory of your Java installation
 - *C:\Program Files\Java\jdk-11.0.11\bin*
- Open a command prompt window and go to the bin directory
- Execute the following command
 - *javac Welcome.java*
- If the source code is ok, then javac (the Java compiler) will produce a file called Welcome.class in the current directory

Compiling a Java Program

- If the source file contains multiple classes then javac will produce separate .class files for each class
- Every compiled class in Java will have their own .class file
- .class files contain the bytecodes of each class
- So, a .class file in Java contains the bytecodes of a single class only

Executing a Java Program

- After successful compilation execute the following command
 - ***java Welcome***
 - *Note that we have omitted the .class extension here*
- The JVM will look for the class file *Welcome.class* and search for a *public static void main(String args[])* method inside the class
- If the JVM finds the above two, it will execute the body of the main method, otherwise it will generate an error and will exit immediately

Another Java Program

The image shows a Java code editor with a file named 'A.java'. The code defines a class 'A' with a constructor, a setter method, and a getter method. It also contains a main method that creates an object of class 'A', sets its value to 10, and prints it out.

```
1 public class A {  
2     private int a;  
3  
4     public A()  
5     {  
6         this.a = 0;  
7     }  
8  
9     public void setA(int a)  
10    {  
11        this.a = a;  
12    }  
13  
14    public int getA()  
15    {  
16        return this.a;  
17    }  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
5510  
5511  
5512  
5513  
5514  
5515  
5516  
5517  
5518  
5519  
5520  
5521  
5522  
5523  
5524  
5525  
5526  
5527  
5528  
5529  
5530  
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5539  
55310  
55311  
55312  
55313  
55314  
55315  
55316  
55317  
55318  
55319  
55320  
55321  
55322  
55323  
55324  
55325  
55326  
55327  
55328  
55329  
55330  
55331  
55332  
55333  
55334  
55335  
55336  
55337  
55338  
55339  
55340  
55341  
55342  
55343  
55344  
55345  
55346  
55347  
55348  
55349  
55350  
55351  
55352  
55353  
55354  
55355  
55356  
55357  
55358  
55359  
55360  
55361  
55362  
55363  
55364  
55365  
55366  
55367  
55368  
55369  
55370  
55371  
55372  
55373  
55374  
55375  
55376  
55377  
55378  
55379  
55380  
55381  
55382  
55383  
55384  
55385  
55386  
55387  
55388  
55389  
55390  
55391  
55392  
55393  
55394  
55395  
55396  
55397  
55398  
55399  
553100  
553101  
553102  
553103  
553104  
553105  
553106  
553107  
553108  
553109  
553110  
553111  
553112  
553113  
553114  
553115  
553116  
553117  
553118  
553119  
553120  
553121  
553122  
553123  
553124  
553125  
553126  
553127  
553128  
553129  
553130  
553131  
553132  
553133  
553134  
553135  
553136  
553137  
553138  
553139  
553140  
553141  
553142  
553143  
553144  
553145  
553146  
553147  
553148  
553149  
553150  
553151  
553152  
553153  
553154  
553155  
553156  
553157  
553158  
553159  
553160  
553161  
553162  
553163  
553164  
553165  
553166  
553167  
553168  
553169  
553170  
553171  
553172  
553173  
553174  
553175  
553176  
553177  
553178  
553179  
553180  
553181  
553182  
553183  
553184  
553185  
553186  
553187  
553188  
553189  
553190  
553191  
553192  
553193  
553194  
553195  
553196  
553197  
553198  
553199  
553200  
553201  
553202  
553203  
553204  
553205  
553206  
553207  
553208  
553209  
553210  
553211  
553212  
553213  
553214  
553215  
553216  
553217  
553218  
553219  
553220  
553221  
553222  
553223  
553224  
553225  
553226  
553227  
553228  
553229  
553230  
553231  
553232  
553233  
553234  
553235  
553236  
553237  
553238  
553239  
553240  
553241  
553242  
553243  
553244  
553245  
553246  
553247  
553248  
553249  
553250  
553251  
553252  
553253  
553254  
553255  
553256  
553257  
553258  
553259  
553260  
553261  
553262  
553263  
553264  
553265  
553266  
553267  
553268  
553269  
553270  
553271  
553272  
553273  
553274  
553275  
553276  
553277  
553278  
553279  
553280  
553281  
553282  
553283  
553284  
553285  
553286  
553287  
553288  
553289  
553290  
553291  
553292  
553293  
553294  
553295  
553296  
553297  
553298  
553299  
5532100  
5532101  
5532102  
5532103  
5532104  
5532105  
5532106  
5532107  
5532108  
5532109  
5532110  
5532111  
5532112  
5532113  
5532114  
5532115  
5532116  
5532117  
5532118  
5532119  
5532120  
5532121  
5532122  
5532123  
5532124  
5532125  
5532126  
5532127  
5532128  
5532129  
5532130  
5532131  
5532132  
5532133  
5532134  
5532135  
5532136  
5532137  
5532138  
5532139  
5532140  
5532141  
5532142  
5532143  
5532144  
5532145  
5532146  
5532147  
5532148  
5532149  
5532150  
5532151  
5532152  
5532153  
5532154  
5532155  
5532156  
5532157  
5532158  
5532159  
5532160  
5532161  
5532162  
5532163  
5532164  
5532165  
5532166  
5532167  
5532168  
5532169  
5532170  
5532171  
5532172  
5532173  
5532174  
5532175  
5532176  
5532177  
5532178  
5532179  
5532180  
5532181  
5532182  
5532183  
5532184  
5532185  
5532186  
5532187  
5532188  
5532189  
5532190  
5532191  
5532192  
5532193  
5532194  
5532195  
5532196  
5532197  
5532198  
5532199  
5532200  
5532201  
5532202  
5532203  
5532204  
5532205  
5532206  
5532207  
5532208  
5532209  
5532210  
5532211  
5532212  
5532213  
5532214  
5532215  
5532216  
5532217  
5532218  
5532219  
5532220  
5532221  
5532222  
5532223  
5532224  
5532225  
5532226  
5532227  
5532228  
5532229  
55322210  
55322211  
55322212  
55322213  
55322214  
55322215  
55322216  
55322217  
55322218  
55322219  
55322220  
55322221  
55322222  
55322223  
55322224  
55322225  
55322226  
55322227  
55322228  
55322229  
55322230  
55322231  
55322232  
55322233  
55322234  
55322235  
55322236  
55322237  
55322238  
55322239  
55322240  
55322241  
55322242  
55322243  
55322244  
55322245  
55322246  
55322247  
55322248  
55322249  
55322250  
55322251  
55322252  
55322253  
55322254  
55322255  
55322256  
55322257  
55322258  
55322259  
55322260  
55322261  
55322262  
55322263  
55322264  
55322265  
55322266  
55322267  
55322268  
55322269  
55322270  
55322271  
55322272  
55322273  
55322274  
55322275  
55322276  
55322277  
55322278  
55322279  
55322280  
55322281  
55322282  
55322283  
55322284  
55322285  
55322286  
55322287  
55322288  
55322289  
55322290  
55322291  
55322292  
55322293  
55322294  
55322295  
55322296  
55322297  
55322298  
55322299  
553222100  
553222101  
553222102  
553222103  
553222104  
553222105  
553222106  
553222107  
553222108  
553222109  
553222110  
553222111  
553222112  
553222113  
553222114  
553222115  
553222116  
553222117  
553222118  
553222119  
553222120  
553222121  
553222122  
553222123  
553222124  
553222125  
553222126  
553222127  
553222128  
553222129  
553222130  
553222131  
553222132  
553222133  
553222134  
553222135  
553222136  
553222137  
553222138  
553222139  
553222140  
553222141  
553222142  
553222143  
553222144  
553222145  
553222146  
553222147  
553222148  
553222149  
553222150  
553222151  
553222152  
553222153  
553222154  
553222155  
553222156  
553222157  
553222158  
553222159  
553222160  
553222161  
553222162  
553222163  
553222164  
553222165  
553222166  
553222167  
553222168  
553222169  
553222170  
553222171  
553222172  
553222173  
553222174  
553222175  
553222176  
553222177  
553222178  
553222179  
553222180  
553222181  
553222182  
553222183  
553222184  
553222185  
553222186  
553222187  
553222188  
553222189  
553222190  
553222191  
553222192  
553222193  
553222194  
553222195  
553222196  
553222197  
553222198  
553222199  
553222200  
553222201  
553222202  
553222203  
553222204  
553222205  
553222206  
553222207  
553222208  
553222209  
553222210  
553222211  
553222212  
553222213  
553222214  
553222215  
553222216  
553222217  
553222218  
553222219  
553222220  
553222221  
553222222  
553222223  
553222224  
553222225  
553222226  
553222227  
553222228  
553222229  
553222230  
553222231  
553222232  
553222233  
553222234  
553222235  
553222236  
553222237  
553222238  
553222239  
553222240  
553222241  
553222242  
553222243  
553222244  
553222245  
553222246  
553222247  
553222248  
553222249  
553222250  
553222251  
553222252  
553222253  
553222254  
553222255  
553222256  
553222257  
553222258  
553222259  
553222260  
553222261  
553222262  
553222263  
553222264  
553222265  
553222266  
553222267  
553222268  
553222269  
553222270  
553222271  
553222272  
553222273  
553222274  
553222275  
553222276  
553222277  
553222278  
553222279  
553222280  
553222281  
553222282  
553222283  
553222284  
553222285  
553222286  
553222287  
553222288  
553222289  
553222290  
553222291  
553222292  
553222293  
553222294  
553222295  
553222296  
553222297  
553222298  
553222299  
553222300  
553222301  
553222302  
553222303  
553222304  
553222305  
553222306  
553222307  
553222308  
553222309  
553222310  
553222311  
553222312  
553222313  
553222314  
553222315  
553222316  
553222317  
553222318  
553222319  
553222320  
553222321  
553222322  
553222323  
553222324  
553222325  
553222326  
553222327  
553222328  
553222329  
553222330  
553222331  
553222332  
553222333  
553222334  
553222335  
553222336  
553222337  
553222338  
553222339  
553222340  
553222341  
553222342  
553222343  
553222344  
553222345  
553222346  
553222347  
553222348  
553222349  
553222350  
553222351  
553222352  
553222353  
553222354  
553222355  
553222356  
553222357  
553222358  
553222359  
553222360  
553222361  
553222362  
553222363  
553222364  
553222365  
553222366  
553222367  
553222368  
553222369  
553222370  
553222371  
553222372  
553222373  
553222374  
553222375  
553222376  
553222377  
553222378  
553222379  
553222380  
553222381  
553222382  
553222383  
553222384  
553222385  
553222386  
553222387  
553222388  
553222389  
553222390  
553222391  
553222392  
553222393  
553222394  
553222395  
553222396  
553222397  
553222398  
553222399  
553222400  
553222401  
553222402  
553222403  
553222404  
553222405  
553222406  
553222407  
553222408  
553222409  
553222410  
553222411  
553222412  
553222413  
553222414  
553222415  
553222416  
553222417  
553222418  
553222419  
553222420  
553222421  
553222422  
553222423  
553222424  
553222425  
553222426  
553222427  
553222428  
553222429  
553222430  
553222431  
553222432  
553222433  
553222434  
553222435  
553222436  
553222437  
553222438  
553222439  
553222440  
553222441  
553222442  
553222443  
553222444  
553222445  
553222446  
553222447  
553222448  
553222449  
553222450  
553222451  
553222452  
553222453  
553222454  
553222455  
553222456  
553222457  
553222458  
553222459  
553222460  
553222461  
553222462  
553222463  
553222464  
553222465  
553222466  
553222467  
553222468  
553222469  
553222470  
553222471  
553222472  
553222473  
553222474  
553222475  
553222476  
553222477  
553222478  
553222479  
553222480  
553222481  
553222482  
553222483  
553222484  
553222485  
553222486  
553222487  
553222488  
553222489  
553222490  
553222491  
553222492  
553222493  
553222494  
553222495  
553222496  
553222497  
553222498  
553222499  
553222500  
553222501  
553222502  
553222503  
553222504  
553222505  
553222506  
553222507  
553222508  
553222509  
553222510  
553222511  
553222512  
553222513  
553222514  
553222515  
553222516  
553222517  
553222518  
553222519  
553222520  
553222521  
553222522  
553222523  
553222524  
553222525  
553222526  
553222527  
553222528  
553222529  
553222530  
553222531  
553222532  
553222533  
553222534  
553222535  
553222536  
553222537  
553222538  
553222539<br
```

Examining A.java

- The variable of a class type is called a reference
 - *ob* is a reference to A object
- Declaring a class reference is not enough, we have to use new to create an object
- Every Java object has to be instantiated using keyword **new**
- We access a public member of a class using the dot operator (.)
 - Dot (.) is the only member access operator in Java
 - Java does not have ->, & and *



Primitive (built-in) Data types

- Integers
 - byte 8-bit integer (new)
 - short 16-bit integer
 - int 32-bit signed integer
 - long 64-bit signed integer
- Real Numbers
 - float 32-bit floating-point number
 - double 64-bit floating-point number
- Other types
 - char 16-bit, Unicode 2.1 character
 - boolean true or false, *false is not 0 in Java*

Boolean Type

```
1  public class Boolean {
2      public static void main(String[] args) {
3          int a = 10;
4          if (a > 0) // if (a) will give compilation error
5          {
6              System.out.println("Inside If");
7          }
8          boolean b = false;
9          if (b)
10         {
11             System.out.println("Inside If");
12         }
13         else
14         {
15             System.out.println("Inside Else");
16         }
17     }
18 }
19
```

Non-primitive Data types

- The non-primitive data types in java are
 - Objects
 - Array
- Non-primitive types are also called reference types

```
public class Box {  
    int L, W, H;  
  
    Box(int l, int w, int h)  
    {  
        L = l;  
        W = w;  
        H = h;  
    }  
  
    public static void main(String[] args)  
    {  
        Box p; // p is a reference pointing to null  
        p = new Box( l: 1, w: 2, h: 3); // now the actual object is created  
    }  
}
```

Primitive vs. Non-primitive type

- Primitive types are handled by value – the actual primitive values are stored in variable and passed to methods

```
int x = 10;  
public MyPrimitive(int x) { }
```

- Non-primitive data types (objects and arrays) are handled by reference – the reference is stored in variable and passed to methods

```
Box b = new Box(1,2,3);  
public MyNonPrimitive(Box x) { }
```

Primitive vs. Non-primitive type

- Primitive types are handled by value
 - There is no easy way to swap two primitive integers in Java
 - No method like **void swap(int *x, int *y)**
 - Can only be done using object or array
- But do we actually need a method to swap?
 - **x += (y - (y = x))** does the same in a single statement

Java References

- Java references are used to point to Java objects created by new
- Java objects are **always** passed **by reference** to other functions, *never by value*
- Java references act as pointers but does not allow pointer arithmetic
- We cannot read the value of a reference and hence cannot find the address of a Java object
- We cannot take the address of a Java reference

Java References

- We can make a Java reference point to a new object
 - By copying one reference to another
ClassName ref2 = ref1; // Here ref1 is declared earlier
 - By creating a new object and assign it to the reference
ClassName ref1 = new ClassName();
- We cannot place arbitrary values to a reference except the special value **null** which means that the reference is pointing to nothing

ClassName ref1 = 100; // compiler error

ClassName ref2 = null; // no problem

Java References

The screenshot shows a Java code editor with the file `Box.java` open. The code defines a class `Box` with three integer fields `L`, `W`, and `H`. It has a constructor that initializes these fields. The `main` method creates two `Box` objects, `b1` and `b2`, both initially pointing to `null`. Then, it creates a new object with dimensions `(8, 5, 7)` and assigns it to `b1`. Both `b1` and `b2` now point to the same object. Next, it creates a new object with dimensions `(3, 9, 2)` and assigns it to `b1`. Now, `b1` points to the new object, and `b2` still points to the original object, which is now discarded.

```
public class Box {
    int L, W, H;

    Box(int l, int w, int h)
    {
        L = l;
        W = w;
        H = h;
    }

    public static void main(String[] args)
    {
        Box b1; // b1 refers to null
        Box b2; // b2 refers to null
        b1 = new Box( l: 8, w: 5, h: 7); // b1 refers to new object (8, 5, 7)
        b2 = b1; // b2 refers to b1, so both refers (8, 5, 7)
        b1 = new Box( l: 3, w: 9, h: 2); // b1 refers to new object (3, 9, 2)
        b1 = b2; // b1 refers to b2, what happens to object (3, 9, 2)
    }
}
```

Java version, IDE, and Textbook

- We will follow Java SE 11 (LTS), latest release - 11.0.11
 - <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>
- We will use IntelliJ IDEA Community version
 - <https://www.jetbrains.com/idea/download/>
- Books
 - Java: The Complete Reference, 11th Edition by Herbert Schildt
 - Effective Java, 3rd edition by Joshua Bloch (for future)