Lecture 2

Java SE – Programming

presentation

Java Programming – Software App Development

Assoc. Prof. Cristian Toma Ph.D.

D.I.C.E/D.E.I.C – Department of Economic Informatics & Cybernetics
www.dice.ase.ro

cristian.toma@ie.ase.ro – Business Card

Agenda for Lecture 2 – Summary of JSE

Java Standard Edition – Uni- & Multi- dimensional Arrays

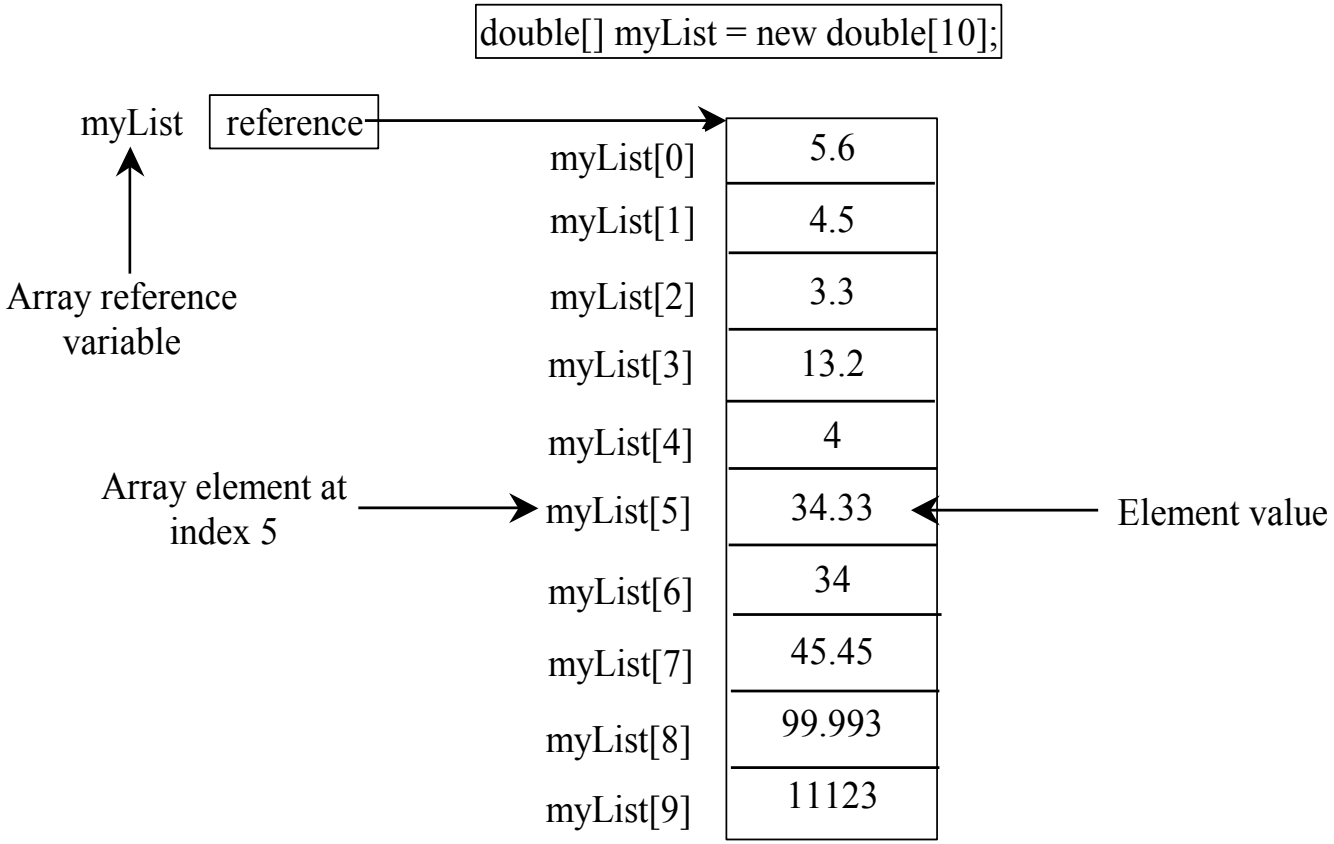
Java Arrays

1.1 Java Arrays – defining and usage

Opening Problem

ASSIGNMENT 03 - Read one hundred numbers, compute their average, and find out how many numbers are above the average.

Array is a data structure that represents a collection of the same types of data.



1.1 Java Arrays – defining and usage

Declaring Array Variables

- `datatype[] arrayRefVar;`

  Example: `double[] myList;`

- `datatype arrayRefVar[];` // This style is allowed, but not preferred

  Example: `double myList[];`

Creating Arrays

```
arrayRefVar = new datatype[arraySize];
```

Example:

```
myList = new double[10];
```

`myList[0]` references the first element in the array.

`myList[9]` references the last element in the array.

1.1 Java Arrays – defining and usage

Declaring, creating, initializing Using the Shorthand Notation

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];

myList[0] = 1.9;

myList[1] = 2.9;

myList[2] = 3.4;

myList[3] = 3.5;
```

This shorthand syntax must be in one statement.

1.1 Java Arrays – defining and usage

Printing arrays

```
for (int i = 0; i < myList.length; i++) {
  System.out.print(myList[i] + " ");
}
```

Summing all elements

```
double total = 0;
for (int i = 0; i < myList.length; i++) {
  total += myList[i];
}
```

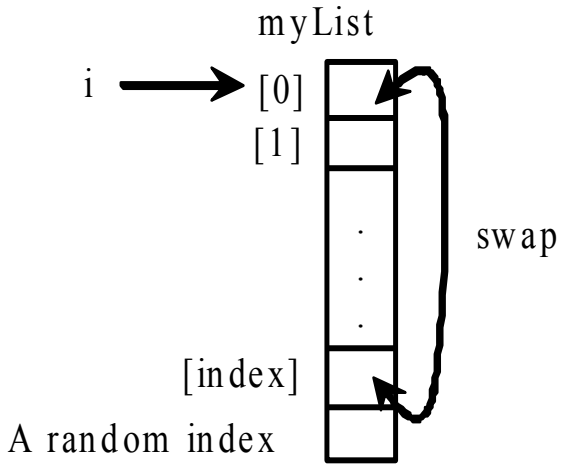
Finding the largest element

```
double max = myList[0];
for (int i = 1; i < myList.length; i++) {
  if (myList[i] > max) max = myList[i];
}
```

1.1 Java Arrays – defining and usage

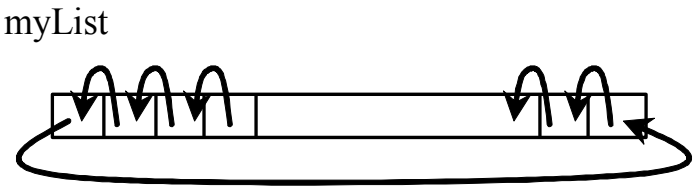
Random shuffling

```
for (int i = 0; i < myList.length; i++) {
  // Generate an index j randomly
  int index = (int)(Math.random()
    * myList.length);

  // Swap myList[i] with myList[j]
  double temp = myList[i];
  myList[i] = myList[index];
  myList[index] = temp;
}
```



Shifting Elements

```
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[myList.length - 1] = temp;
```



1.1 Java Arrays – defining and usage

Enhanced for Loop (for-each loop)

JDK 1.5 introduced a new for loop that enables you to traverse the complete array sequentially without using an index variable. For example, the following code displays all elements in the array *myList*:

```
for (double value : myList)
  System.out.println(value);
```

In general, the syntax is

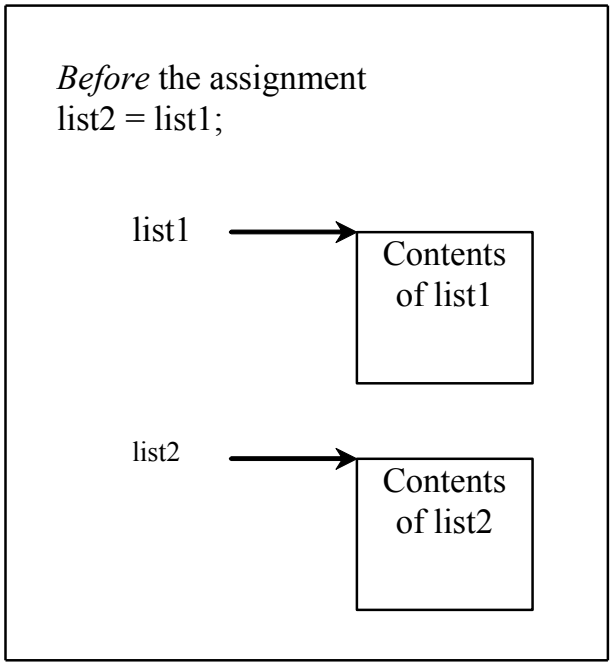
```
for (elementType value : arrayRefVar) {
  // Process the value
}
```

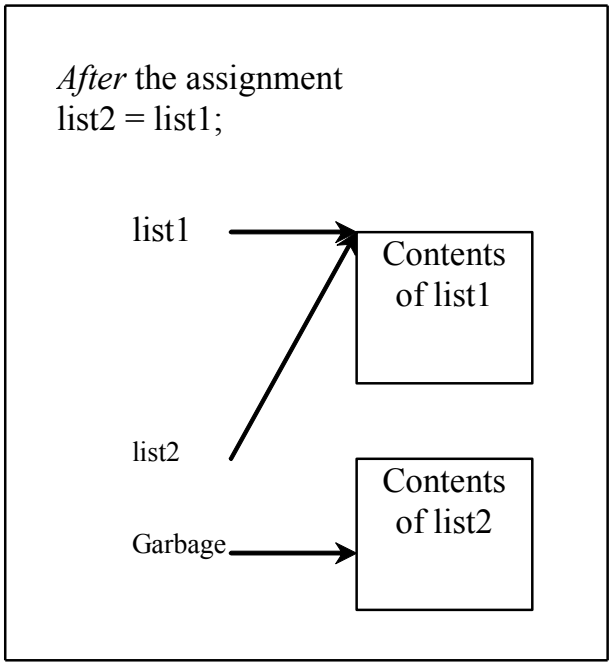
You still have to use an index variable if you wish to traverse the array in a different order or change the elements in the array.

Copying Arrays

Often, in a program, you need to duplicate an array or a part of an array. In such cases you could attempt to use the assignment statement (=), as follows:

list2 = list1;





Copying Arrays

Using a loop:

```
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new int[sourceArray.length];

for (int i = 0; i < sourceArrays.length; i++)
   targetArray[i] = sourceArray[i];
```

The `arraycopy` Utility

```
arraycopy(sourceArray, src_pos, targetArray, tar_pos,
  length);
```

Example:

```
System.arraycopy(sourceArray, 0, targetArray, 0,
  sourceArray.length);
```

1.1 Java Arrays – defining and usage

Passing Arrays to Methods

```
public static void printArray(int[] array)
{
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

```
Invoke the method

int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

```
Invoke the method
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array

Pass By Value

Java uses *pass by value* to pass arguments to a method. There are important differences between passing a value of variables of primitive data types and passing arrays.

• For a parameter of a primitive type value, the actual value is passed. Changing the value of the local parameter inside the method does not affect the value of the variable outside the method.

• For a parameter of an array (or reference) type, the value of the parameter contains a reference to an array / object; this reference is passed to the method. Any changes to the array / object that occur inside the method body will affect the original array / object that was passed as the argument.

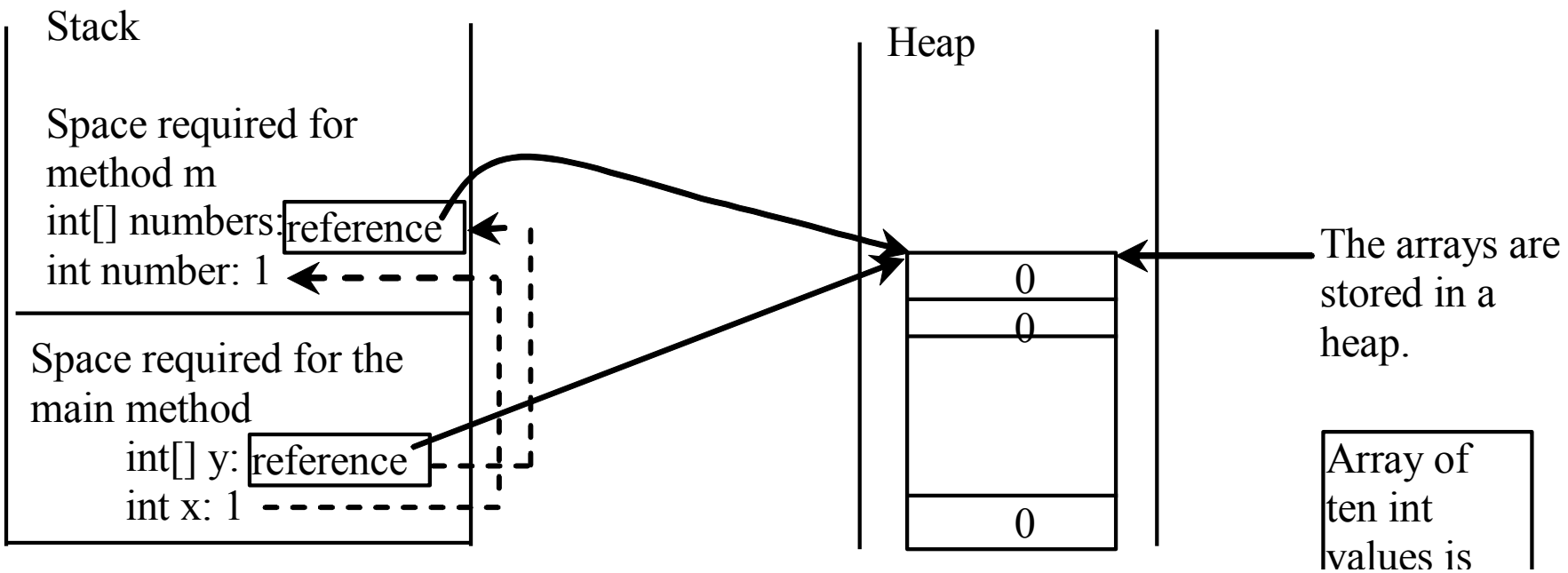
1.1 Java Arrays – defining and usage

Simple Example – Pass by Value

```
public class Test {
  public static void main(String[] args) {
    int x = 1; // x represents an int value
    int[] y = new int[10]; // y represents an array of int
  values

    m(x, y); // Invoke m with arguments x and y

    System.out.println("x is " + x);
    System.out.println("y[0] is " + y[0]);
  }

  public static void m(int number, int[] numbers) {
    number = 1001; // Assign a new value to number
    numbers[0] = 5555; // Assign a new value to numbers[0]
  }
}
```

1.1 Java Arrays – defining and usage

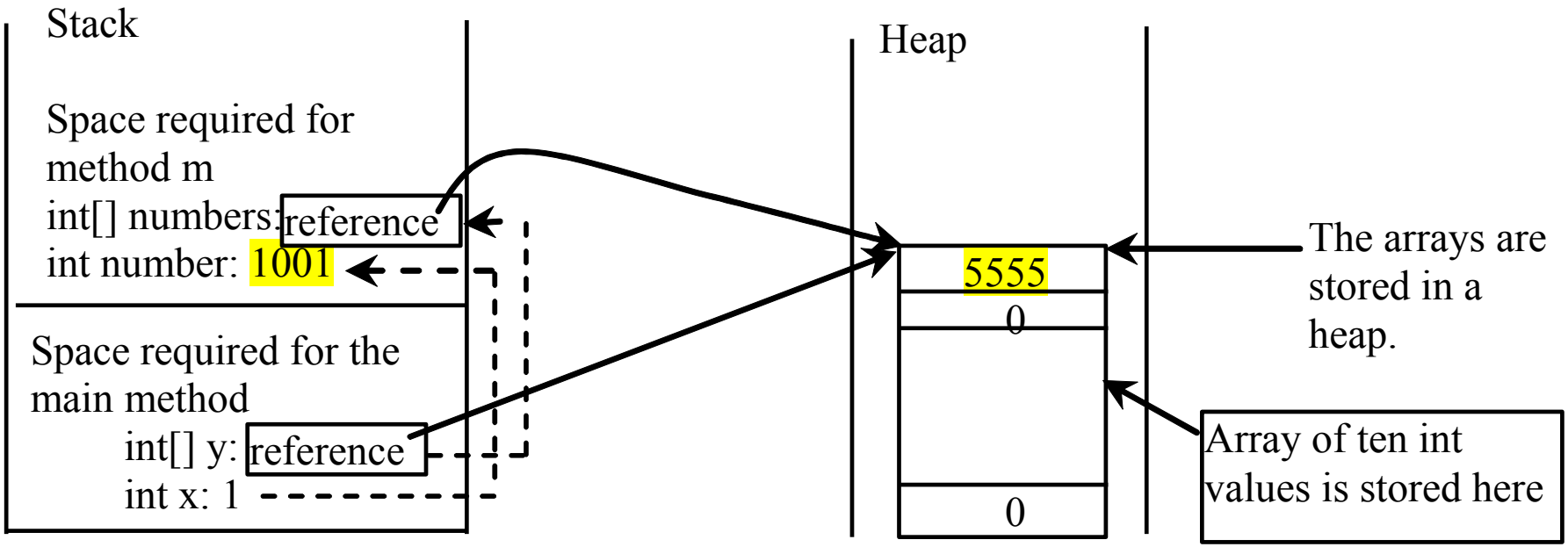
Call Stack – Pass by Value



When invoking m(x, y), the values of x and y are passed to number and numbers. Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

1.1 Java Arrays – defining and usage

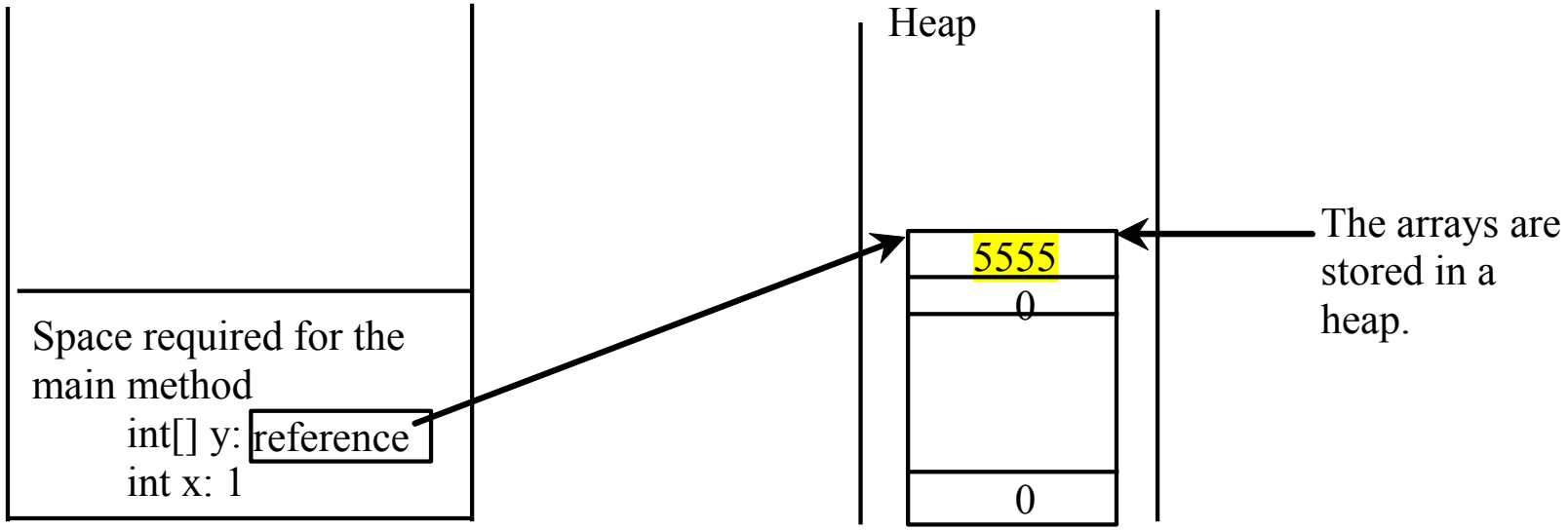
Call Stack – Pass by Value



When invoking m(x, y), the values of x and y are passed to number and numbers. Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

1.1 Java Arrays – defining and usage

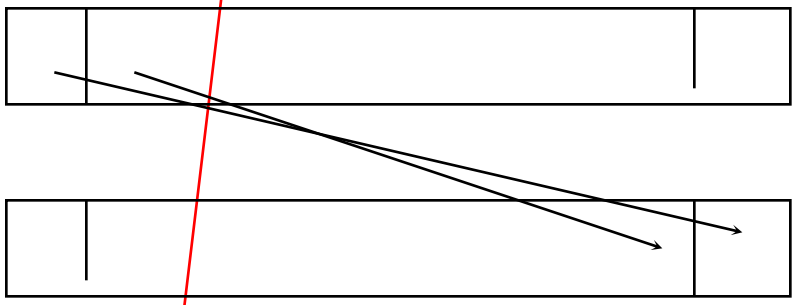
Heap – Pass by Value



The JVM stores the array in an area of memory, called *heap*, which is used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order.

Returning an Array from a Method

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```



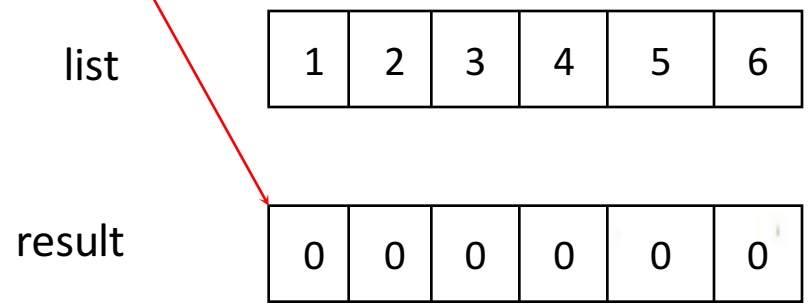

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

Declare result and create array

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

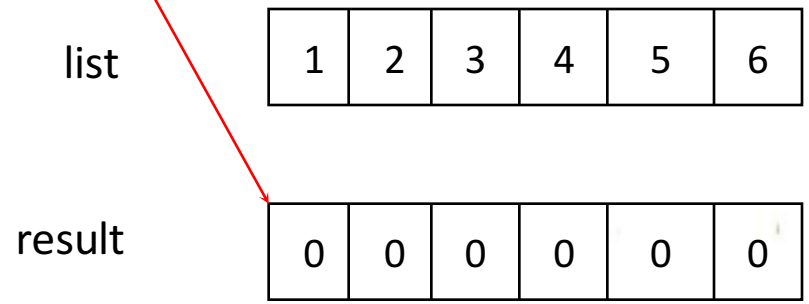


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

i = 0 and j = 5

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

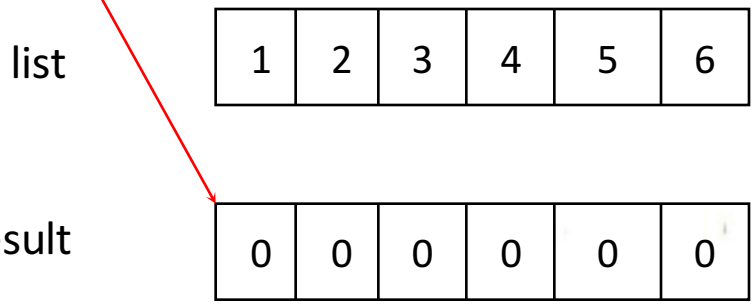


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

i (= 0) is less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

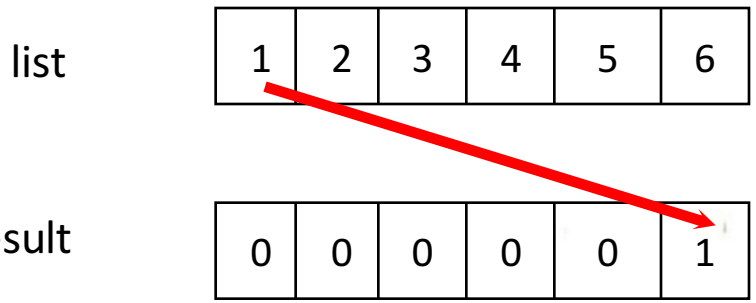


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 0 and j = 5
Assign list[0] to result[5]

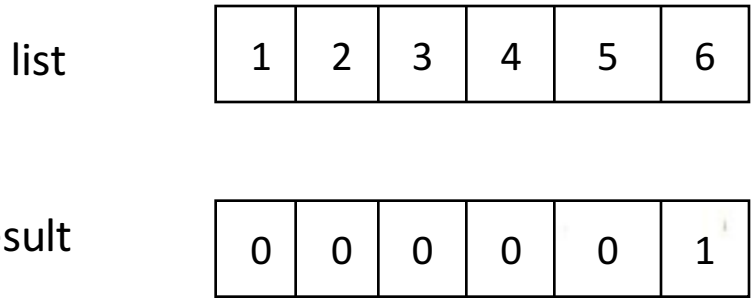


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

After this, i becomes 1 and j becomes 4

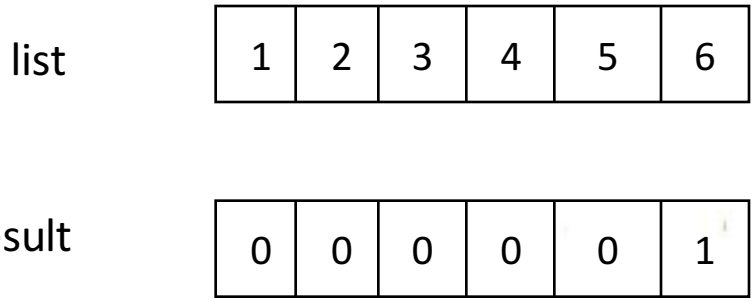


1.5 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i (=1) is less than 6

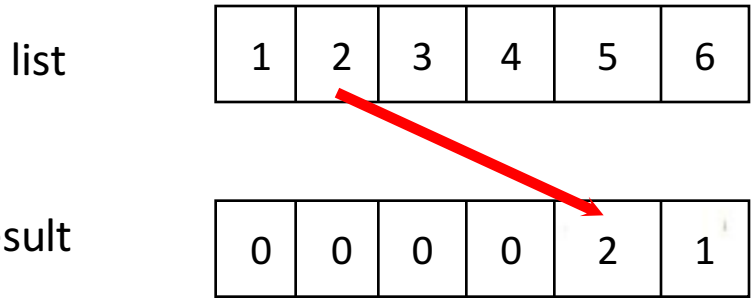


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 1 and j = 4
Assign list[1] to result[4]

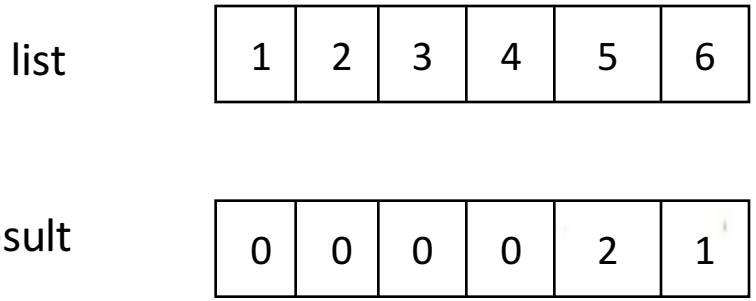


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

After this, i becomes 2 and j becomes 3

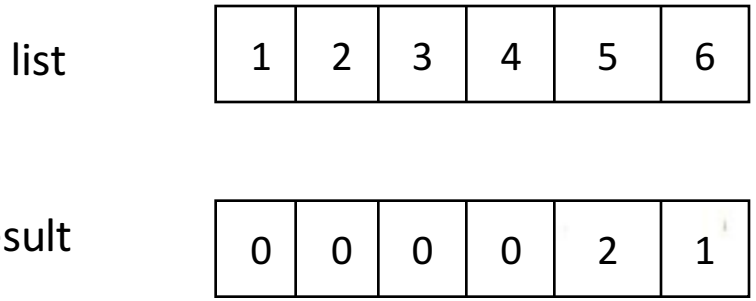


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i (=2) is still less than 6

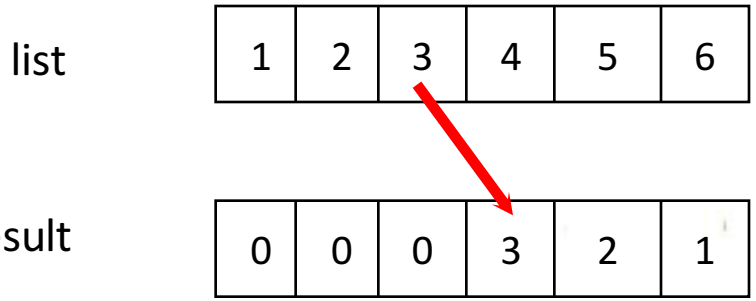


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 2 and j = 3
Assign list[i] to result[j]

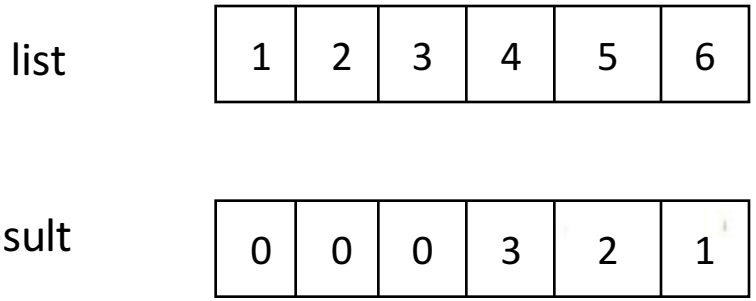


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

After this, i becomes 3 and j becomes 2

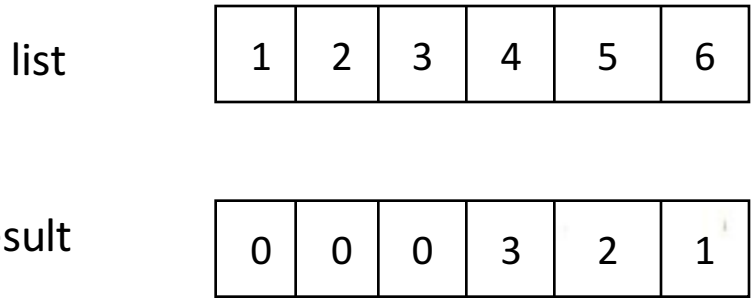


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i (=3) is still less than 6

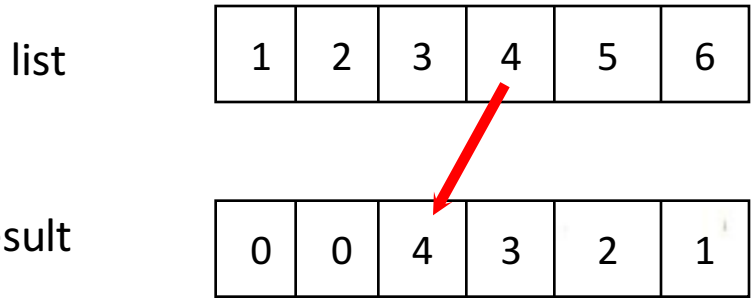


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 3 and j = 2
Assign list[i] to result[j]

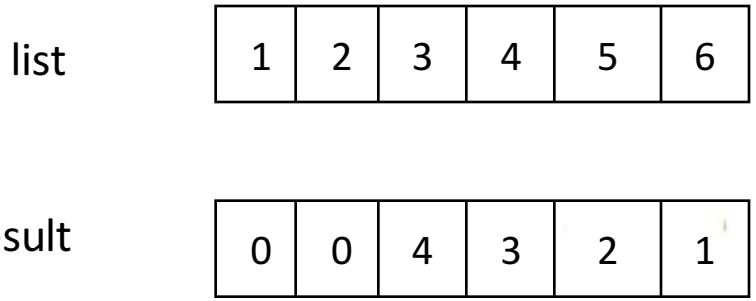


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

After this, i becomes 4 and j becomes 1

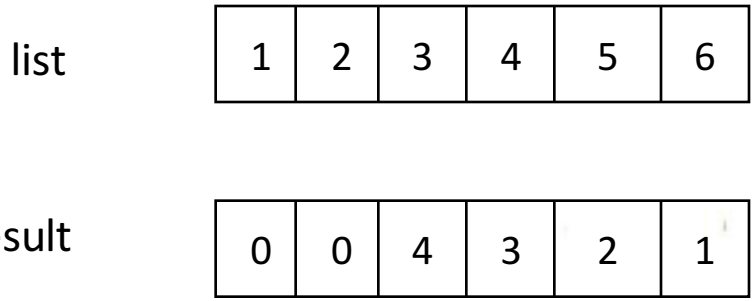


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i (=4) is still less than 6

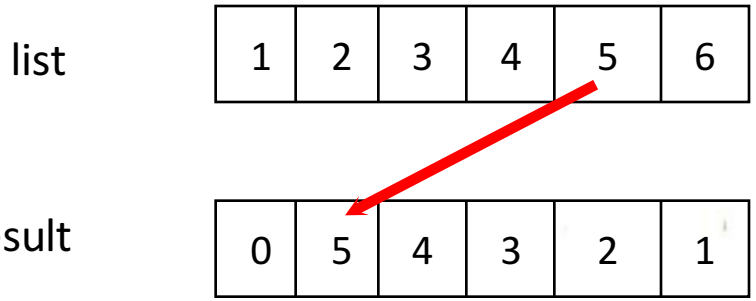


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 4 and j = 1
Assign list[i] to result[j]

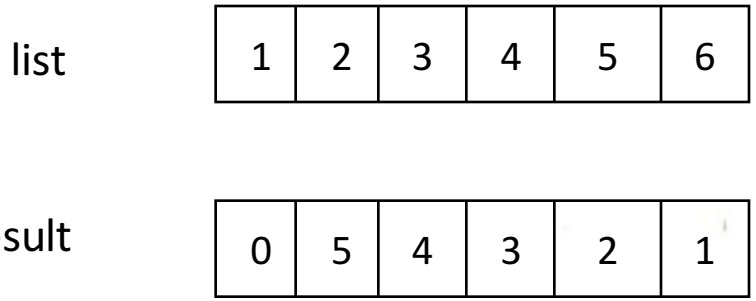


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

After this, i becomes 5 and j becomes 0

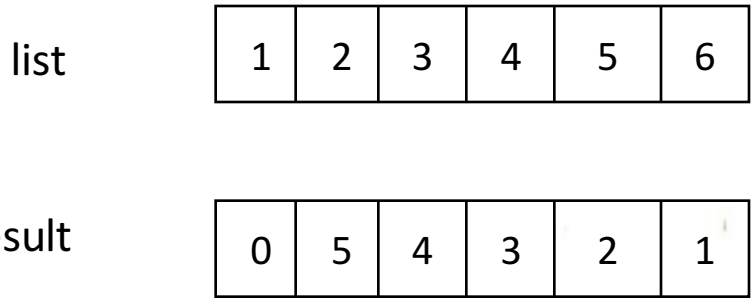


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i (=5) is still less than 6

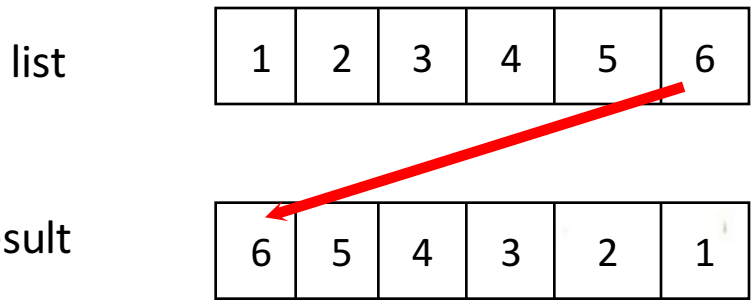


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 5 and j = 0
Assign list[i] to result[j]

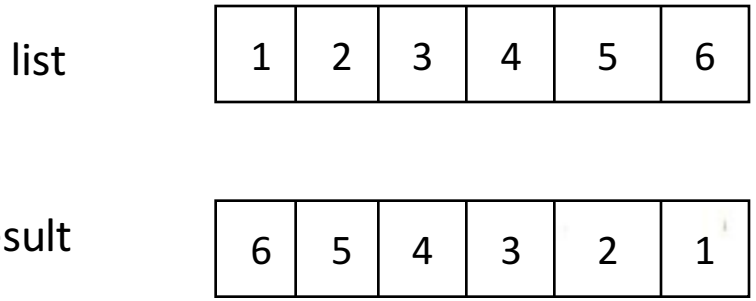


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

After this, i becomes 6 and j becomes -1

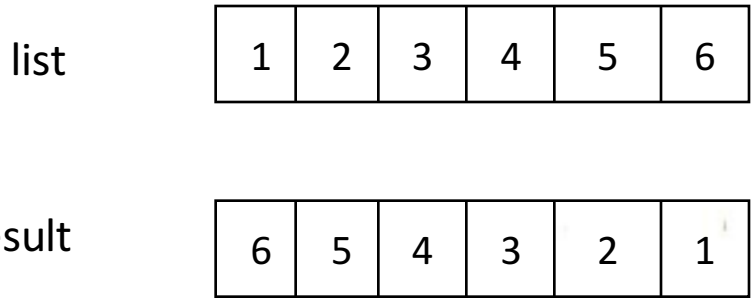


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i (=6) < 6 is false. So exit the loop.

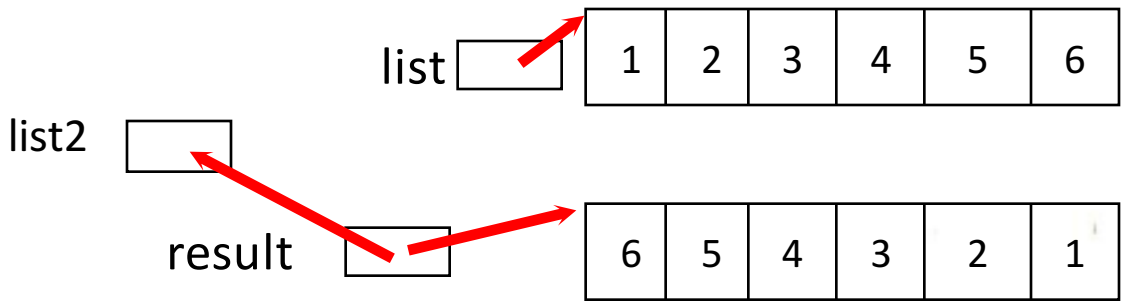


1.1 Java Arrays – Trace the reverse Method

```
int[] list1 = new int[]{1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

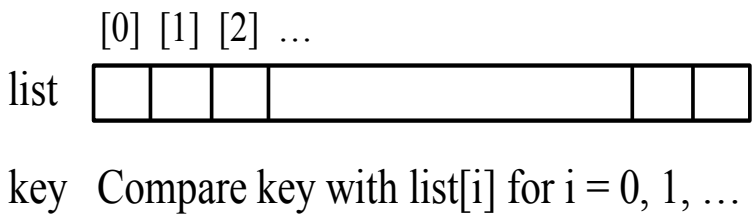
Return result



Searching Arrays

Searching is the process of looking for a specific element in an array; for example, discovering whether a certain score is included in a list of scores. Searching is a common task in computer programming. There are many algorithms and data structures devoted to searching. In this section, two commonly used approaches are discussed, *linear search* and *binary search*.

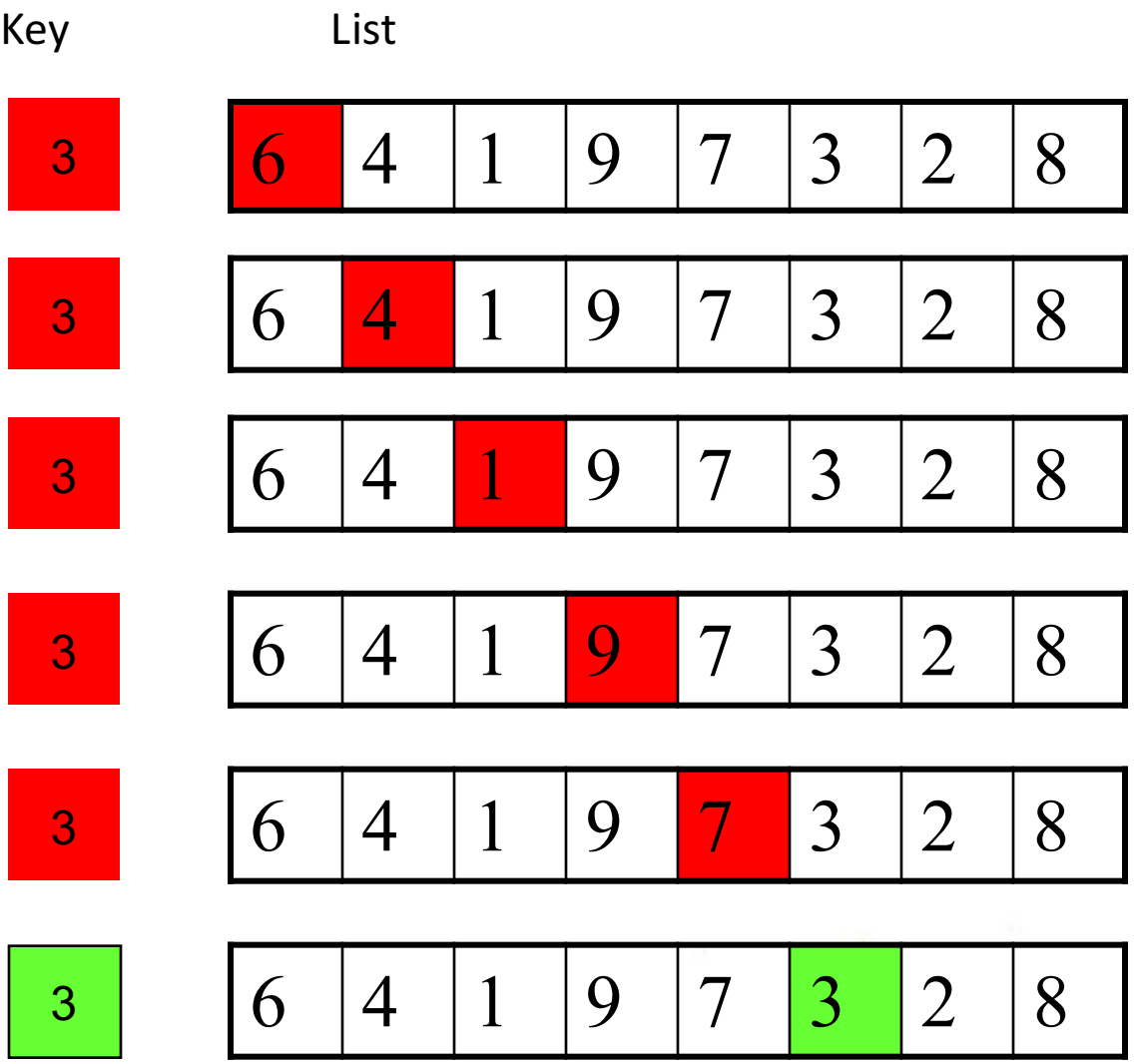
```
public class LinearSearch {
  /** The method for finding a key in the list */
  public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++)
      if (key == list[i])
        return i;
    return -1;
  }
}
```



Linear Search

The linear search approach compares the key element, key, *sequentially* with each element in the array list. The method continues to do so until the key matches an element in the list or the list is exhausted without a match being found. If a match is made, the linear search returns the index of the element in the array that matches the key. If no match is found, the search returns -1.

Linear Search Animation



From Idea to Solution – Linear Search

```
/** The method for finding a key in the list */
public static int linearSearch(int[] list, int key) {
  for (int i = 0; i < list.length; i++)
    if (key == list[i])
      return i;
  return -1;
}
```

Trace the method

```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
int i = linearSearch(list, 4);  // returns 1
int j = linearSearch(list, -4); // returns -1
int k = linearSearch(list, -3); // returns 5
```

Binary Search

For binary search to work, the elements in the array **must already be ordered (SORTED)**. Without loss of generality, assume that the array is in ascending order.

e.g., 2 4 7 10 11 45 50 59 60 66 69 70 79

The binary search first compares the key with the element in the middle of the array.

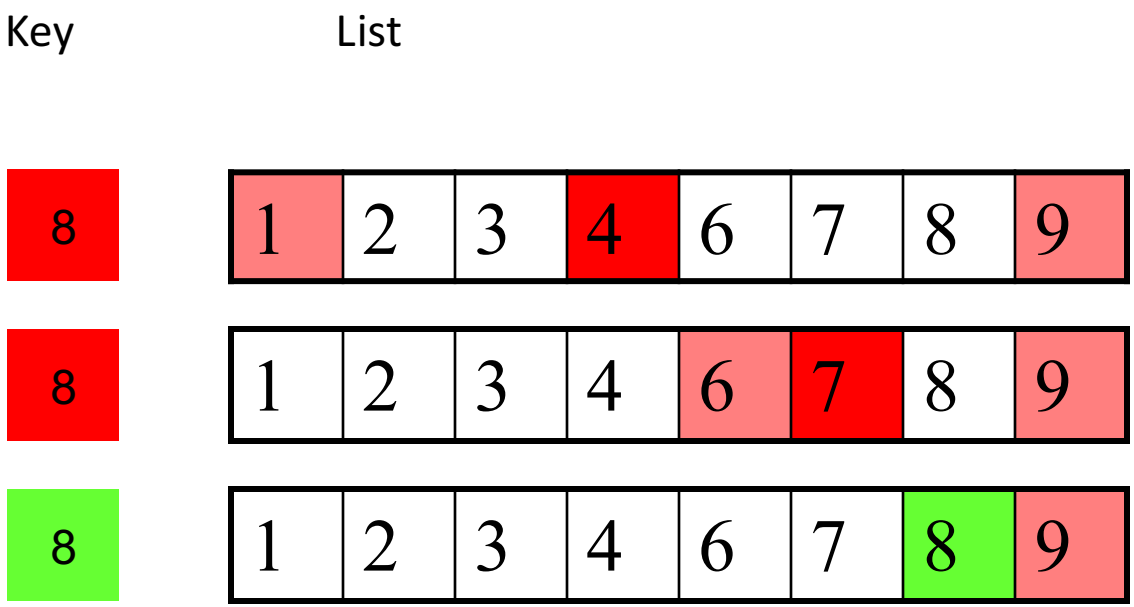
For objects / references a Comparable<E> interface must be implemented in class E.

Binary Search, cont.

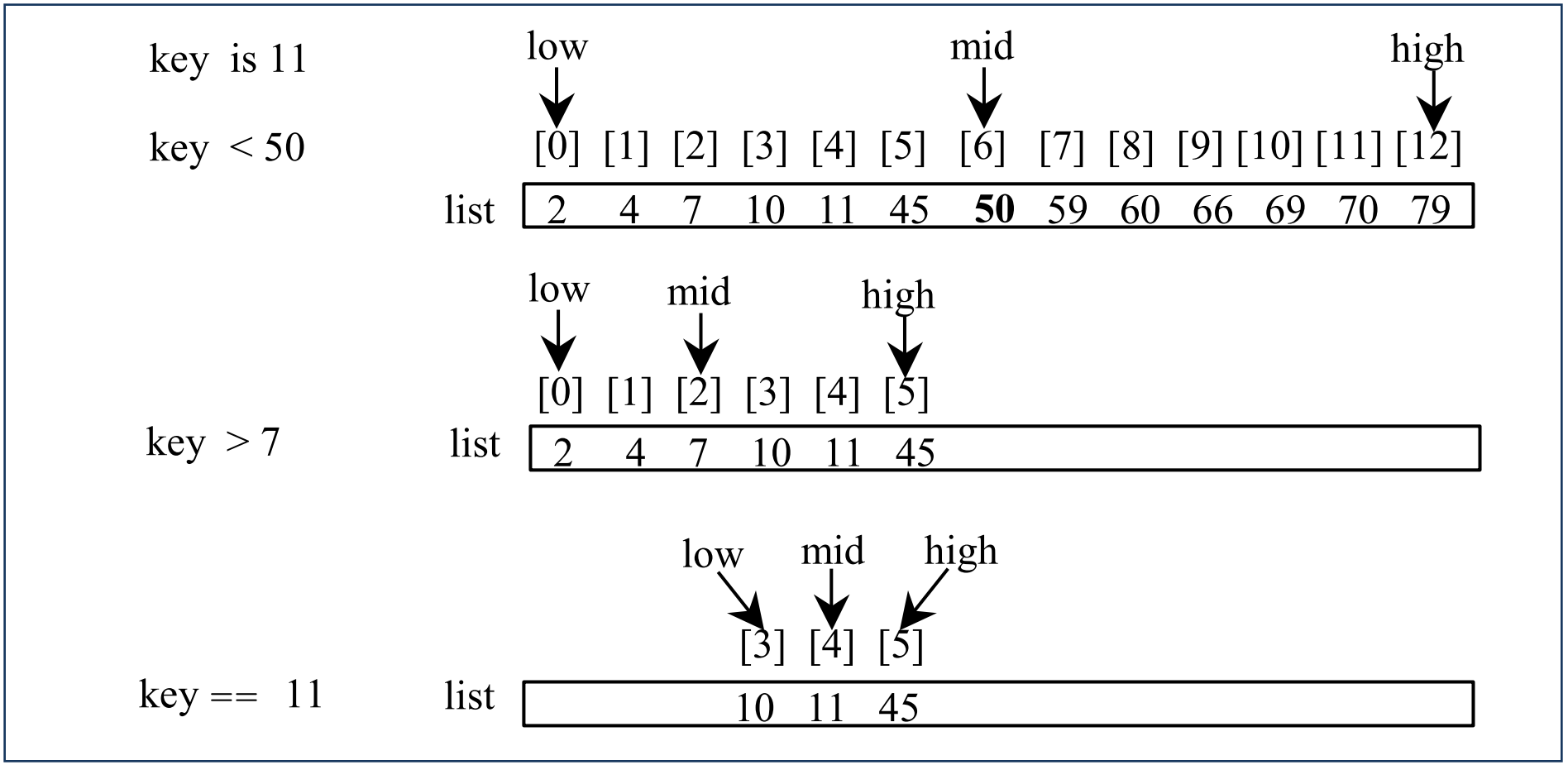
Consider the following three cases:

- If the key is less than the middle element, you only need to search the key in the first half of the array.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, you only need to search the key in the second half of the array.

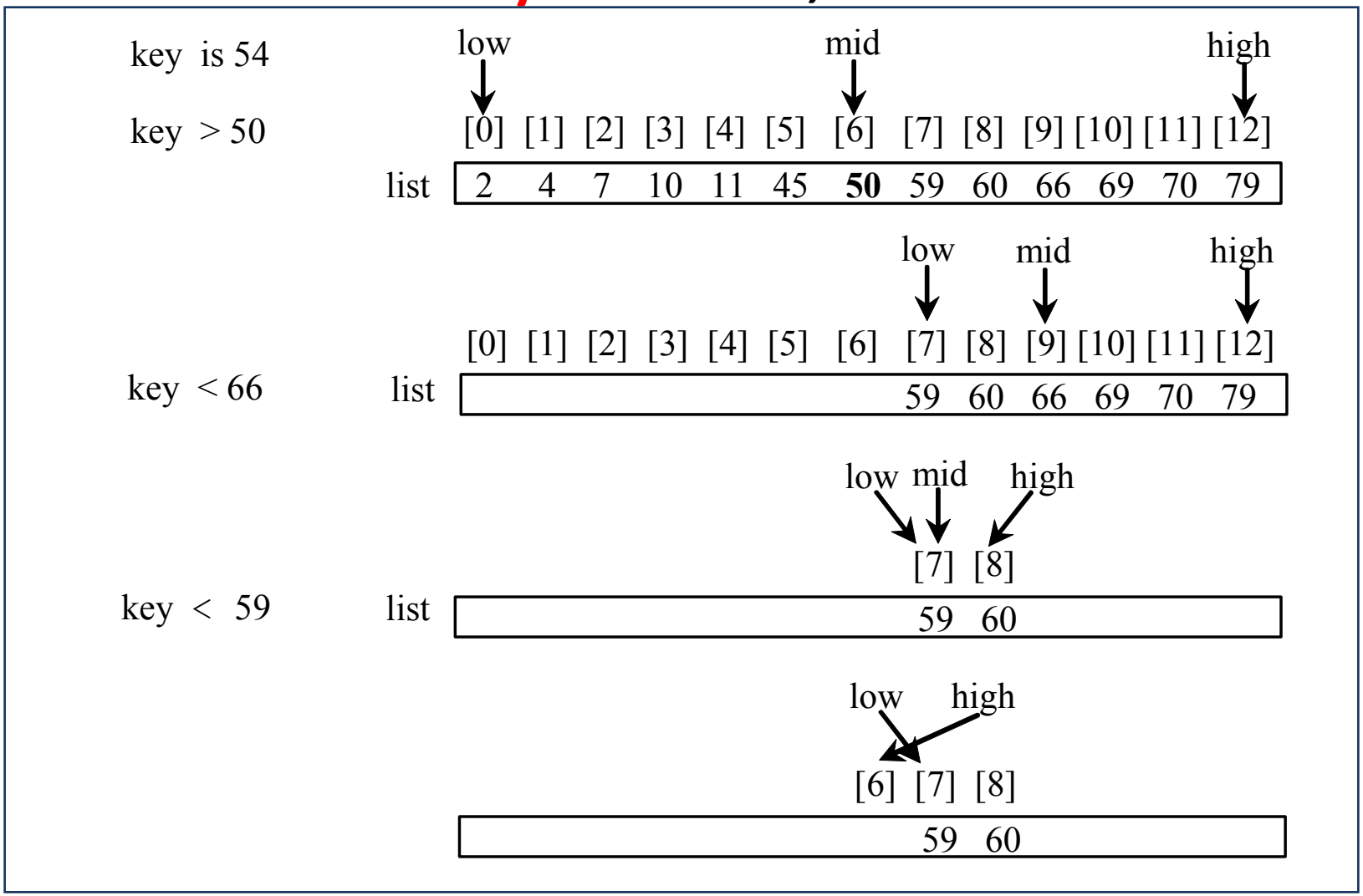
Binary Search



Binary Search, cont.



Binary Search, cont.



Binary Search, cont.

The binarySearch method returns the index of the element in the list that matches the search key if it is contained in the list. Otherwise, it returns

-insertion point - 1.

The insertion point is the point at which the key would be inserted into the list.

From Idea to Solution – Binary Search

```
/** Use binary search to find the key in the list */
public static int binarySearch(int[] list, int key) {
  int low = 0;
  int high = list.length - 1;

  while (high >= low) {
    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;
  }

  return -1 - low;
}
```

The Arrays.binarySearch Method

Since binary search is frequently used in programming, Java provides several overloaded binarySearch methods for searching a key in an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code searches the keys in an array of numbers and an array of characters.

```
int[] list = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};
System.out.println("Index  is " +
  java.util.Arrays.binarySearch(list, 11));

char[] chars = {'a', 'c', 'g', 'x', 'y', 'z'};
System.out.println("Index  is " +
  java.util.Arrays.binarySearch(chars, 't'));
```

Return is 4

Return is –4 (insertion point is 3, so return is -3-1)

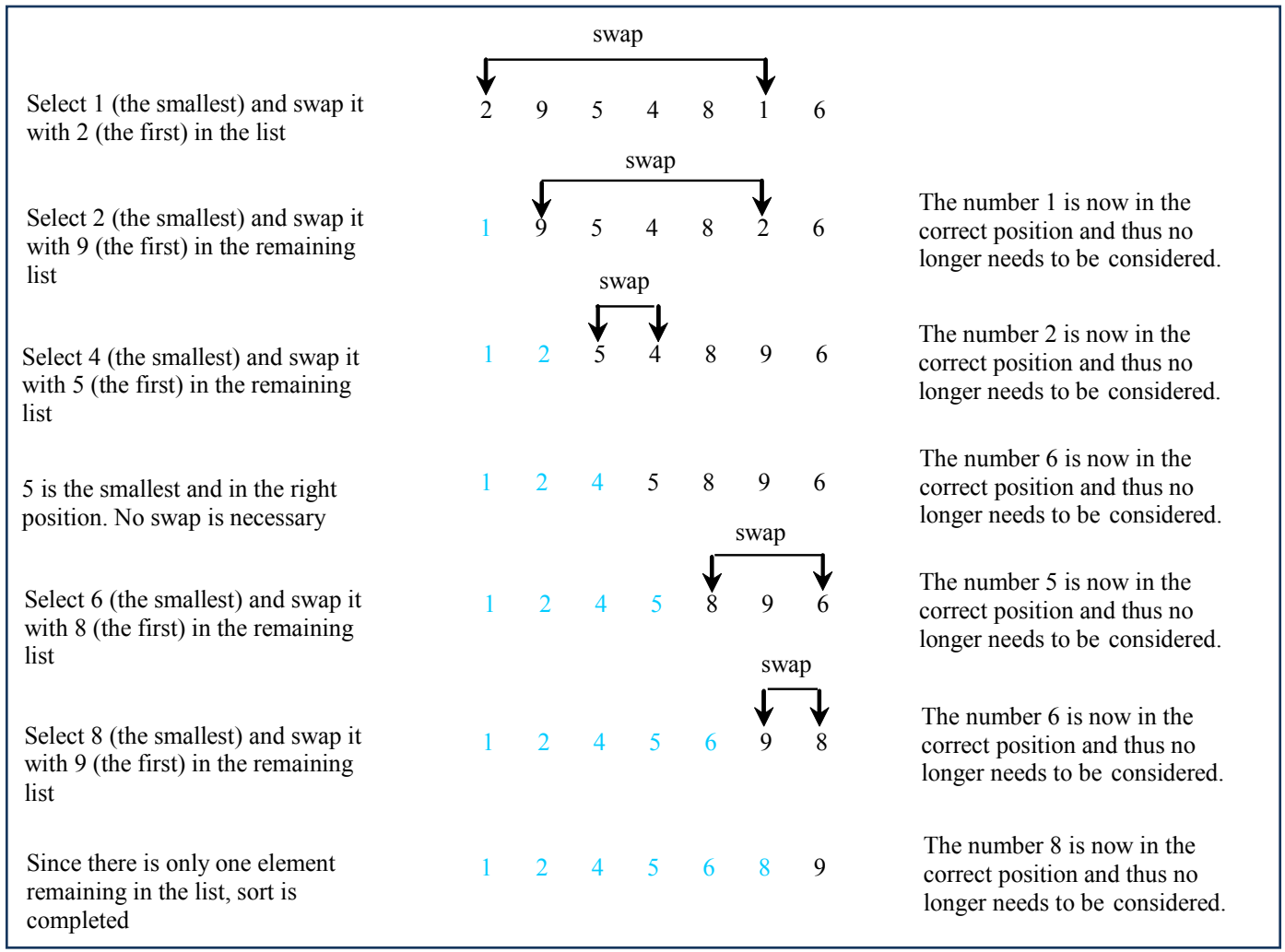
For the binarySearch method to work, the array must be pre-sorted in increasing order.

Sorting Arrays

Sorting, like searching, is also a common task in computer programming. Many different algorithms have been developed for sorting. Mainly are used two simple, intuitive sorting algorithms: *selection sort* and *insertion sort*.

1.1 Java Arrays – Selection Sort

Selection sort finds the largest number in the list and places it last. It then finds the largest number remaining and places it next to last, and so on until the list contains only a single number. Figure shows how to sort the list {2, 9, 5, 4, 8, 1, 6} using selection sort.



1.1 Java Arrays – Selection Sort

From Idea to Solution - Selection Sort

```
for (int i = 0; i < list.length; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

```
list[0] list[1] list[2] list[3] ...                    list[10]

list[0] list[1] list[2] list[3] ...                    list[10]

list[0] list[1] list[2] list[3] ...                    list[10]

list[0] list[1] list[2] list[3] ...                    list[10]

list[0] list[1] list[2] list[3] ...                    list[10]

                                ...

list[0] list[1] list[2] list[3] ...                    list[10]
```

1.5 Java Arrays – Selection Sort

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

Expand

```
double currentMin = list[i];
[                                        ]
for (int j = i; j < list.length; j++) {
  if (currentMin > list[j]) {
    currentMin = list[j];
    [                                    ]
  }
}
```

1.1 Java Arrays – Selection Sort

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

Expand

```
double currentMin = list[i];
int currentMinIndex = i;
for (int j = i; j < list.length; j++) {
  if (currentMin > list[j]) {
    currentMin = list[j];
    currentMinIndex = j;
  }
}
```

1.1 Java Arrays – Selection Sort

```
for (int i = 0; i < listSize; i++)
{
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

Expand

```
if (currentMinIndex != i) {
  list[currentMinIndex] = list[i];
  list[i] = currentMin;
}
```

1.1 Java Arrays – Selection Sort – Wrap it in a Method

```
/** The method for sorting the numbers */
public static void selectionSort(double[] list) {
    for (int i = 0; i < list.length; i++) {
      // Find the minimum in the list[i..list.length-1]
      double currentMin = list[i];
      int currentMinIndex = i;
      for (int j = i + 1; j < list.length; j++) {
        if (currentMin > list[j]) {
          currentMin = list[j];
          currentMinIndex = j;
        }
      }
      // Swap list[i] with list[currentMinIndex] if necessary;
      if (currentMinIndex != i) {
        list[currentMinIndex] = list[i];
        list[i] = currentMin;
      }
    }
  }
```

Invoke it

selectionSort(yourList)

The Arrays.sort Method

Since sorting is frequently used in programming, Java provides several overloaded sort methods for sorting an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code sorts an array of numbers and an array of characters.

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);

char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
java.util.Arrays.sort(chars);
```

1.2 Java Arrays – Multidimensional Arrays

Motivations: Thus far, you have used one-dimensional arrays to model linear collections of elements. You can use a two-dimensional array to represent a matrix or a table. For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

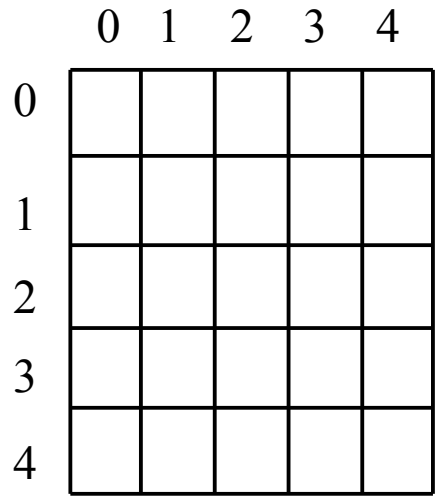
Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

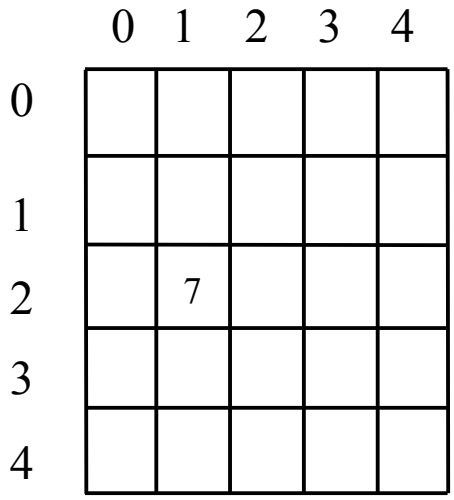
Declare/Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

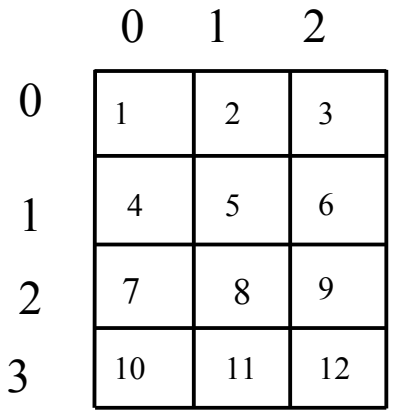
Two-dimensional Array Illustration




```
matrix = new int[5][5];
```




```
matrix[2][1] = 7;
```




```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

1.2 Java Arrays – Multidimensional Arrays

Declaring, Creating, and Initializing Using Shorthand Notations

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

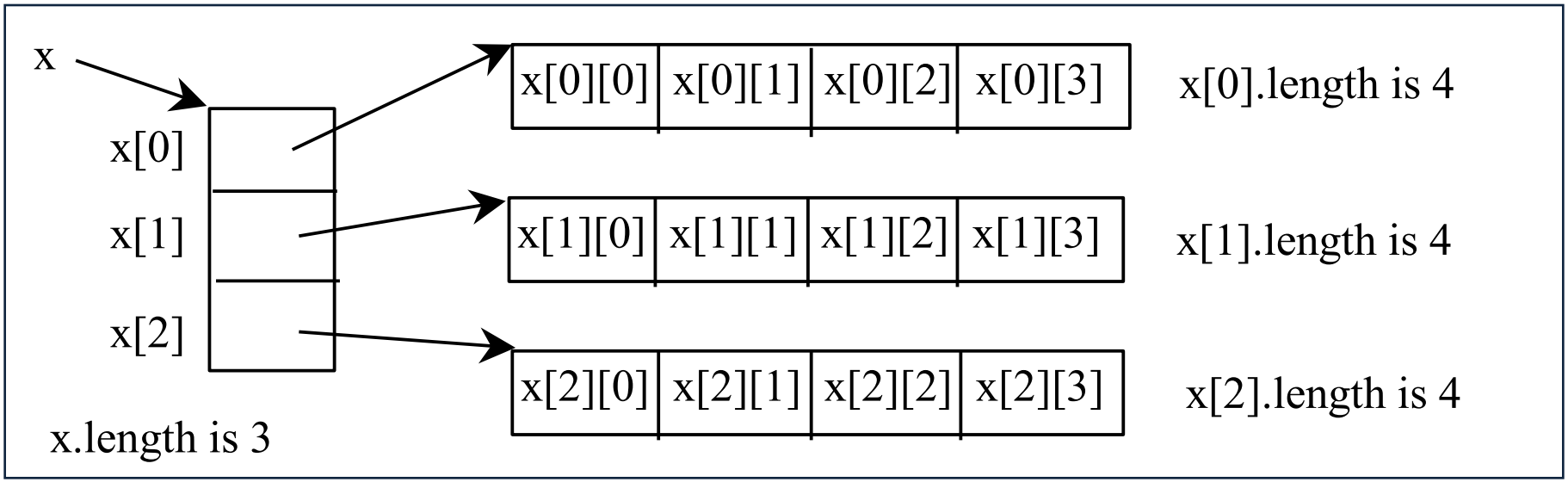
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

Lengths of Two-dimensional Arrays

```
int[][] x = new int[3][4];
```



Ragged Arrays

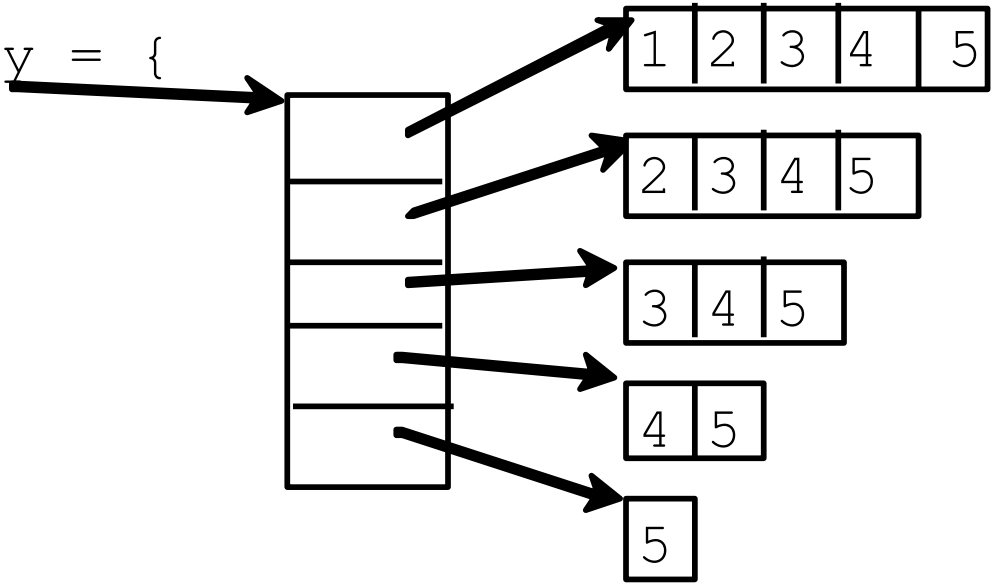
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

Ragged Arrays

```
int[][] triangleArray = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```



Section Conclusion

Fact: **Java is natural**

In few **samples** it is simple to remember: selections, loops, methods, arrays – it is like in C/C++ syntax.

Java OOP

Java OOP & JSE

2. Summary of Java SE - OOP

Analogy with C++ and it will be reloaded in next lecture

What is a class?

What is a package of classes in Java?

What is an object / instance?

How many bytes has an object inside JVM?

Why do we need clone, equals and hash methods?

Demo & memory model for the ***Certificate*** Java class

2. Summary of Java SE - OOP

DEMO: Java Arrays demo and other samples + Java Libraries: JARs + Java 9 or 11 MODULES

Object Oriented Programming
for easy Java sharing

Section Conclusions

Object Oriented Programming in Java – class, object, object's deep vs. shallow copy, interface, interface as type, abstract class, inheritance, polymorphism.

Class is used to encapsulate attributes (fields | features | characteristics) and methods (behavior | 'interface') in order to produce instances / objects.

Object is a instance of a class with certain values for attributes and with same methods class for all the generated instances.

In OOP there are two major relationships: "has a" (composition) and "is a" (inheritance)

Repositories: Git, Build Automation: ANT / Maven / Gradle, DevOps Continous Build Integration: Jenkins, Deployment in Cloud: Docker vs. IaaS VMs

Advanced SW Eng. & DevOps Topics

3.1 Java Library

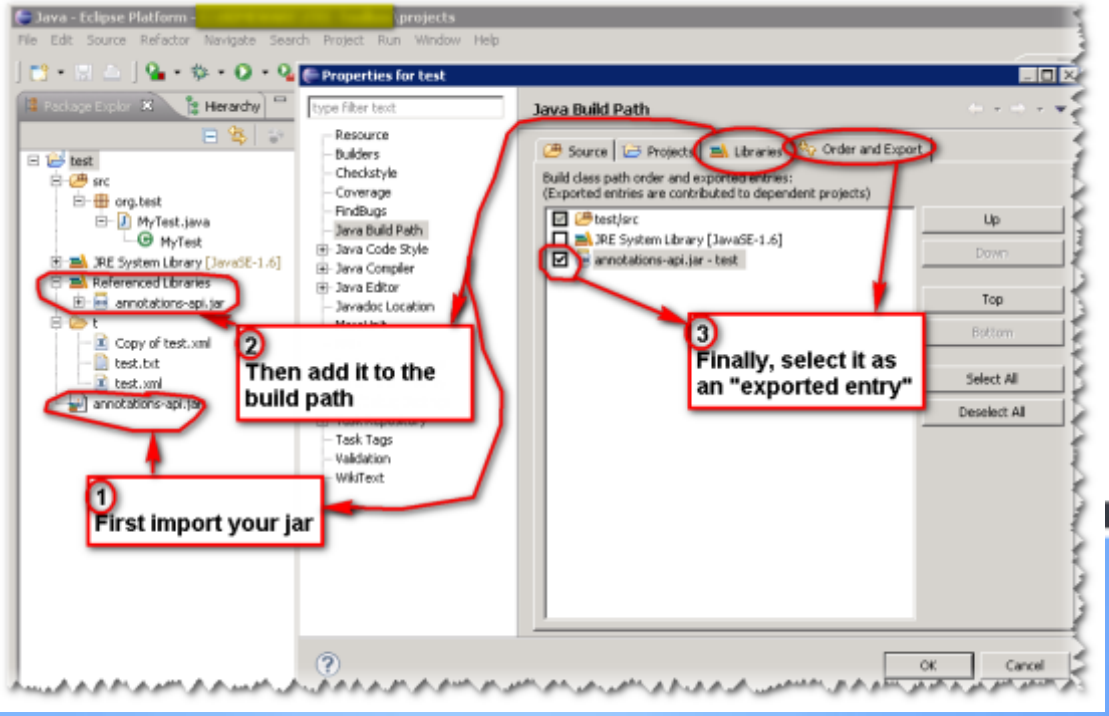
What is a **Java library**?

What are the advantages and disadvantages of Java libraries?

How can be solved in Java multiple dependencies or inclusions of the same class in the compilation phase? How were solved these problems in C/C++?

How should be created a Java library and how should be used – command line vs. IDE?

>jar -cvf archive_name.jar files_names_to_compress
>javac -classpath .:archieve_name.jar *.java
>java –classpath .:archive_name.jar file_with_main_class



3.1 JDK 9 and 11+ Modules

Java 9 Modules:

Java SE 9: Jigsaw Project

Jigsaw project is going to introduce completely new concept of Java SE 9: **Java Module System**. It is very big and prestigious project from Oracle Corp in Java SE 9 release. Initially they have started this project as part of Java SE 7 Release. However with huge changes, it's postponed to Java SE 8 then again postponed. Now it has been released with Java SE 9 in 2017.

Main Goals of Jigsaw Project:

- The Modular JDK - As we know, Current JDK system is too big. So they have decided to divide JDK itself into small modules to get a number of benefits (We will discuss them soon in the coming sections).
- Modular Source Code - Current source code jar files are too big, especially rt.jar is too big right. So they are going to divide Java Source code into smaller modules.
- Modular Run-Time Images - The main goal of this Feature is "Restructure the JDK and JRE run-time images to accommodate modules".
- Encapsulate Most Internal APIs - The main goal of this feature is "Make most of the JDK's internal APIs inaccessible by default but leave a few critical, widely-used internal APIs accessible".
- Java Platform Module System - The main goal of this Feature is "Allowing the user to create their modules to develop their applications".
- jlink: The Java Linker - The main goal of this jlink Tool is "Allowing the user to create executable to their applications".

3.1 JDK 9 and 11+ Modules

Java 9 Modules:

Problems of Current Java 8 System?

In this section, we will discuss about "Why we need Java SE 9 Module System" that means the problems of Current Java System.

Java SE 8 or earlier systems have following problems in developing or delivering Java Based applications.

- As JDK is too big, it is bit tough to scale down to small devices. Java SE 8 has introduced 3 types of compact profiles to solve this problem: compact1, compact2 and compact3. But it does not solve this problem.
- JAR files like rt.jar etc are too big to use in small devices and applications.
- As JDK is too big, our applications or devices are not able to support better Performance.
- There is no Strong Encapsulation in the current Java System because "public" access modifier is too open. Everyone can access it.
- As JDK, JRE is too big, it is hard to Test and Maintain applications.
- As public is too open, They are not to avoid the accessing of some Internal Non-Critical APIs like sun.*, *.internal.* etc.
- As User can access Internal APIs too, Security is also big issue.
- Application is too big.
- Its a bit tough to support Less Coupling between components.

3.1 JDK 9 and 11+ Modules

Java 9 Modules:

Advantages of Java SE 9 Module System

Java SE 9 Module System is going to provide the following benefits:

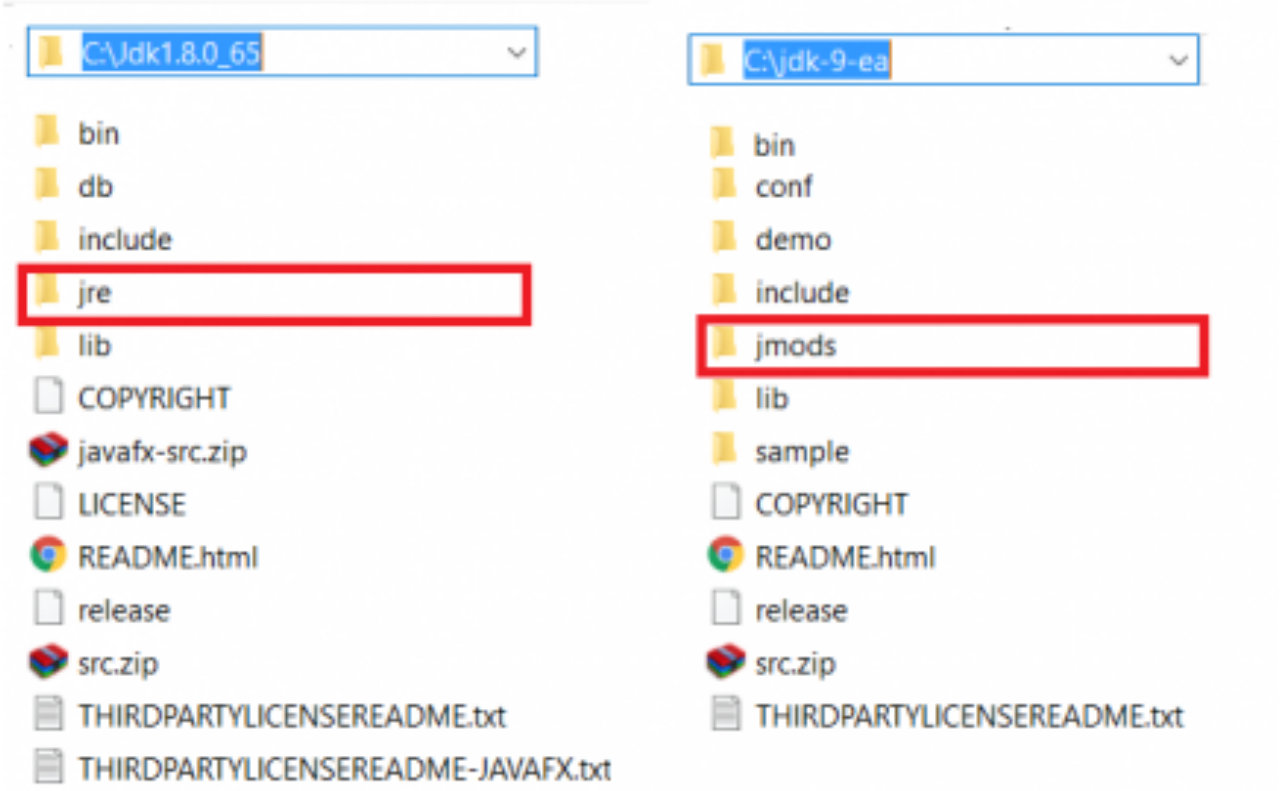
- As Java SE 9 is going to divide JDK, JRE, JARs etc, into smaller modules, we can use whatever modules we want. So it is very easy to scale down the Java Application to Small devices.
- Ease of Testing and Maintainability.
- Supports better Performance.
- As public is not just public, it supports very Strong Encapsulation. (Don't worry its a big concept. we will explore it with some useful examples soon).
- We cannot access Internal Non-Critical APIs anymore.
- Modules can hide unwanted and internal details very safely, we can get better Security.
- Application is too small because we can use only what ever modules we want.
- Its easy to support Less Coupling between components.
- Its easy to support Single Responsibility Principle (SRP).

3.1 JDK 9 and 11+ Modules

Java 9 Modules:

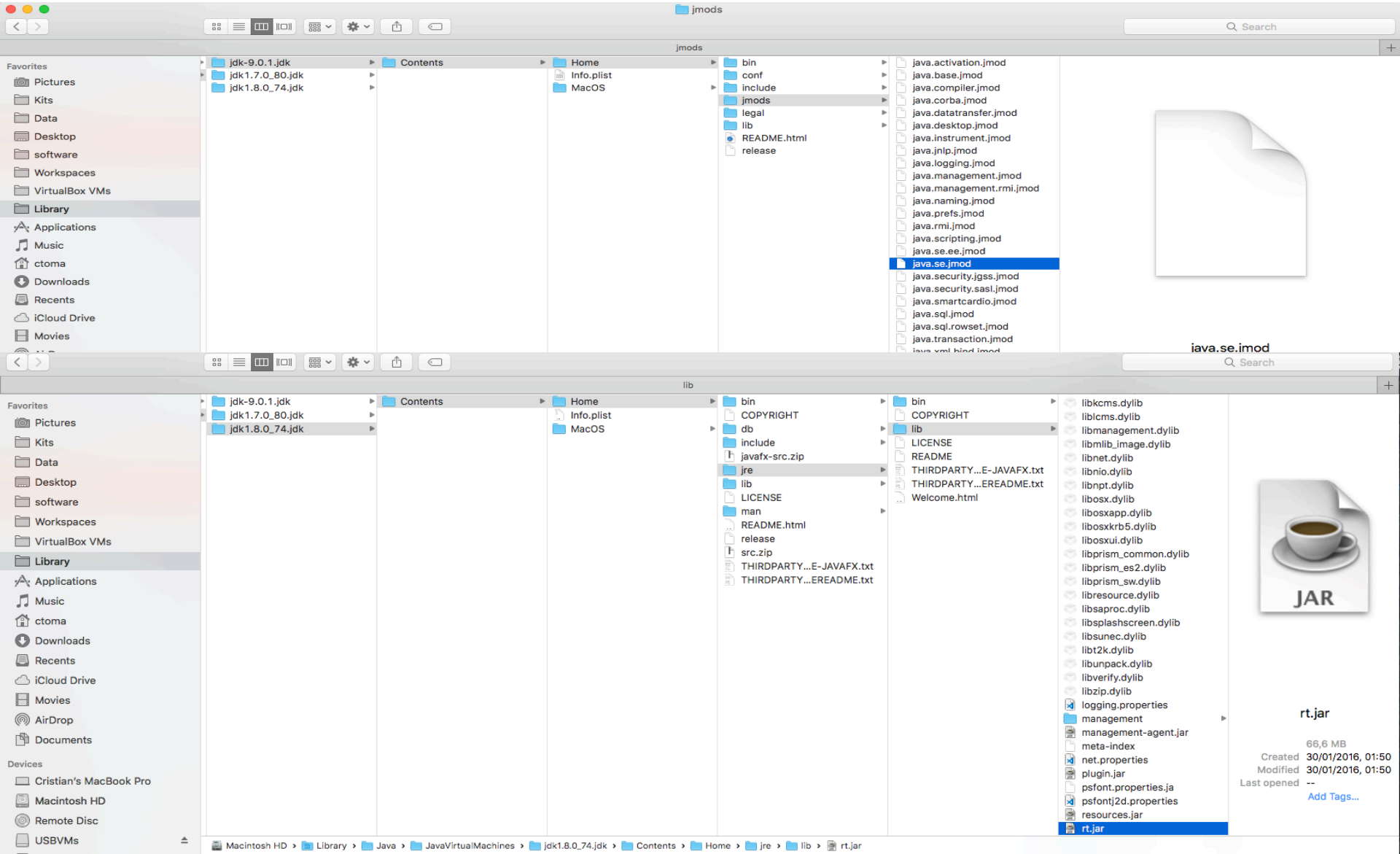
Compare JDK 8 and JDK 9

We know what a JDK software contains. After installing JDK 8 software, we can see couple of directories like bin, jre, lib etc in Java Home folder. However Oracle Corp / OpenJDK has changed this folder structure a bit differently as shown below.



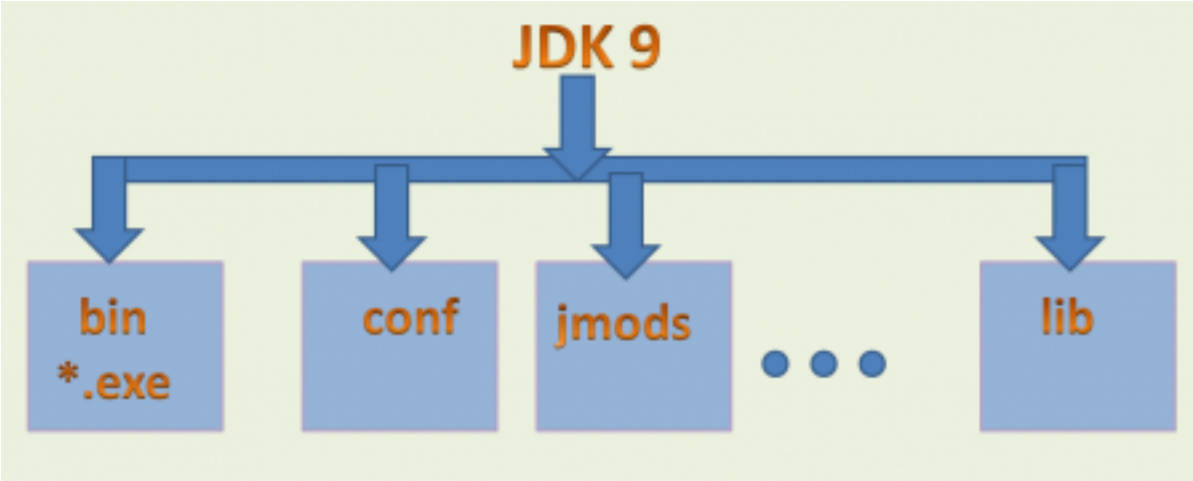
3.1 JDK 9 and 11+ Modules

Java 9 Modules - Compare JDK 8 and JDK 9 Directory Structure:



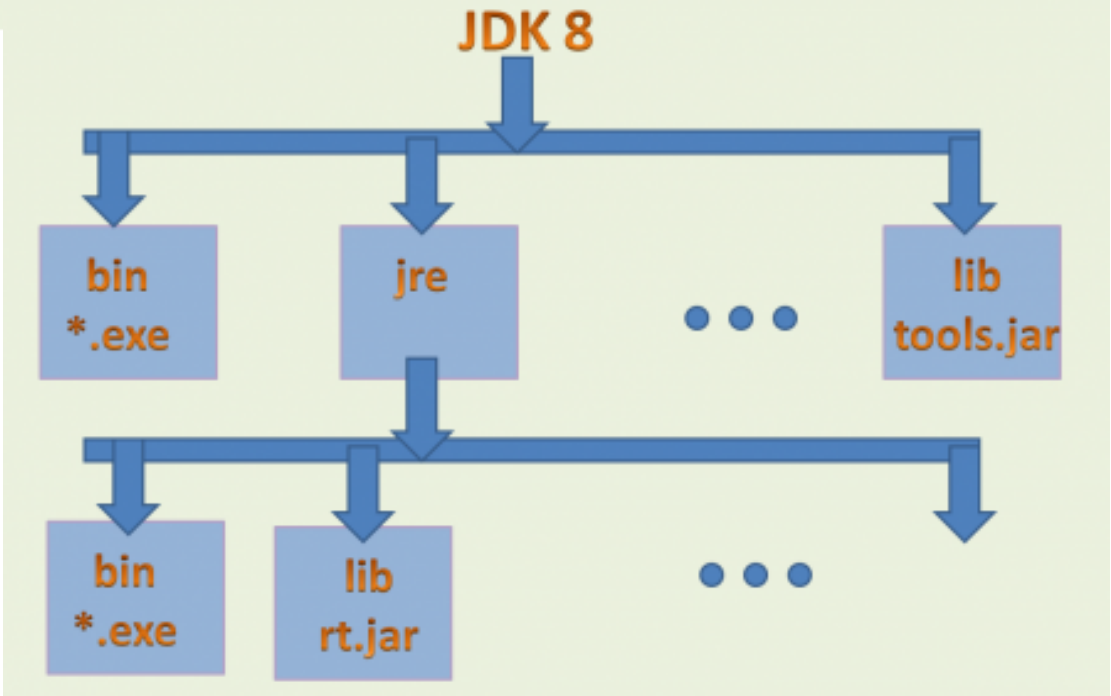
3.1 JDK 9 and 11+ Modules

Java 9 Modules – SDK Distro:



Here JDK 9 does NOT contain JRE.

In JDK 9, JRE is separated into a separate distribution folder. JDK 9 software contains a new folder "jmods". As of today, "jmods" contains 97 modules. The "jmods" folder is available at ${JAVA_HOME}/jmods. These are known as JDK Modules.



3.1 JDK 9 and 11+ Modules

Java 9 Modules:

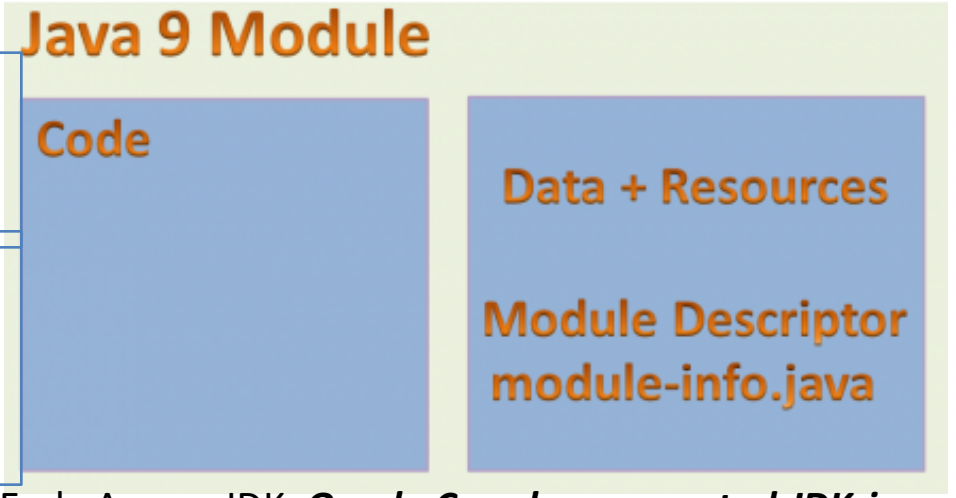
What is Java 9 Module?

A Module is a self-describing collection of Code, Data and some Resources. It is a set of related Packages, Types (classes, abstract classes, interfaces etc) with Code & Data and Resources.

Each Module contains only a set of related code and data to support Single Responsibility (Functionality) Principle (SRP).

The main goal of Java 9 Module System is to support:
Modular Programming in Java.

In Simple Terms,
Module = Code + Data
Module Descriptor ("module-info.java") is a one of the Resources in Java 9 Module.



As of now, Java 9 Module System has 97 modules in Early Access JDK. ***Oracle Corp has separated JDK jars and Java SE Specifications into two set of Modules.***

- ***All JDK Modules starts with "jdk.*"***
- ***All Java SE Specifications Modules starts with "java.*"***

Java 9 Module System has a "java.base" Module. It's known as Base Module. It's an Independent module and does NOT dependent on any other modules. By default, all other Modules dependent on this module.

That's why "java.base" Module is also known as the root of the Java 9 Modules.

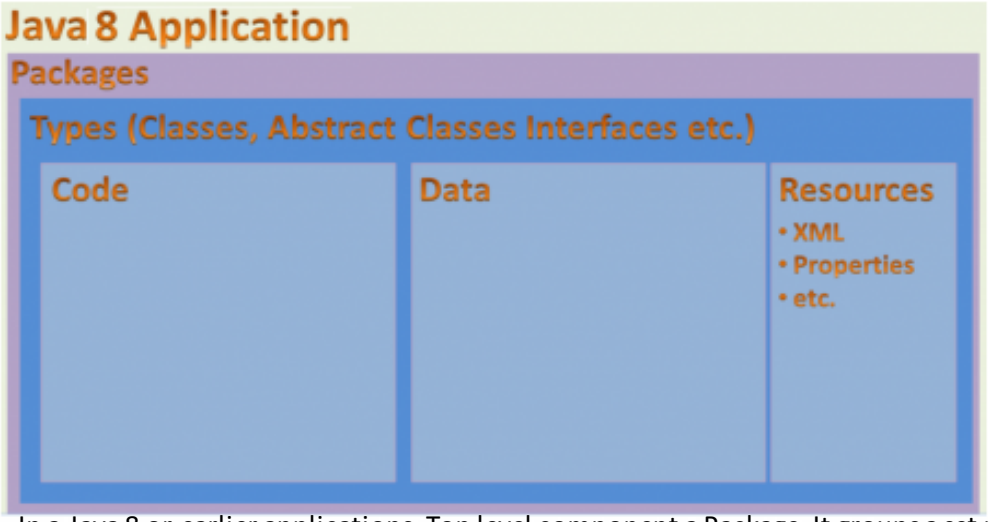
It's default module for all JDK Modules and User-Defined Modules.

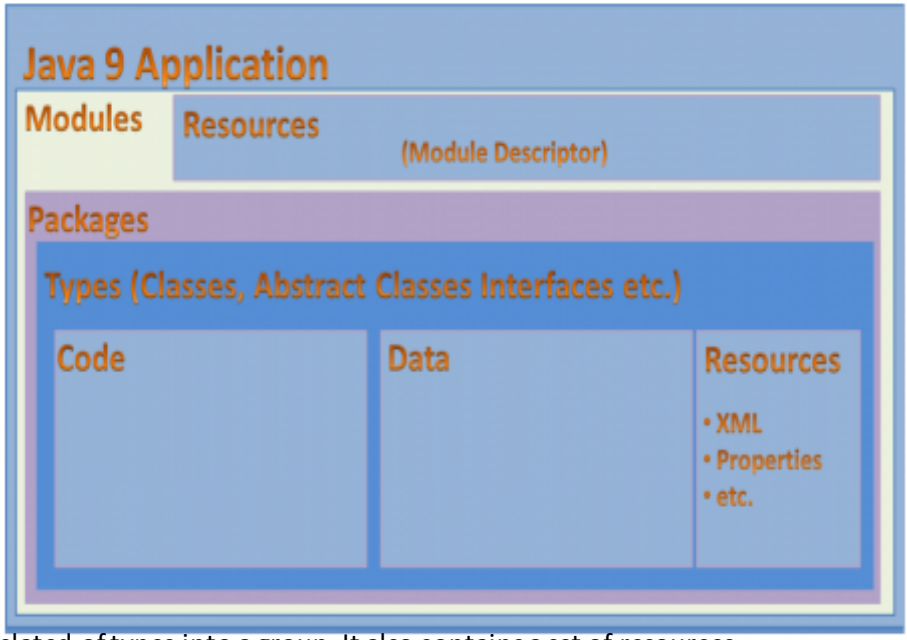
3.1 JDK 9 and 11+ Modules

Java 9 Modules:

Compare Java 8 and Java 9 Applications

We have already developed many Java applications using Java 5, 6,7, or 8. We know how a Java 8 earlier applications looks like and what it contains.





In a Java 8 or earlier applications, Top level component a Package. It groups a set related of types into a group. It also contains a set of resources.

Java 9 Applications does not have much difference with this. It just introduced a new component called "Module", which is used to group a set of related Packages into a group. And one more new component that **Module Descriptor ("module-info.java").** That's it.

Like Java 8 applications have Packages as a Top level components, Java 9 applications have Module as Top Level components.

NOTE: -Each Java 9 Module have one and only one Module and one Module Descriptor. Unlike Java 8 Packages, We cannot create multiple modules into a single Module. In brief we can say a Java 9 Module contains the following main components:

- One Module + Module Name
- Module Descriptor
- Set of Packages
- Set of Types and Resources

Here Resources may be **module-info.java (Module Descriptor)** or any other properties or XML.

3.1 JDK 9 and 11+ Modules

Java 9 Modules:

The two main goals of Java 9 Module System:

- Reliable Configuration
- Strong Encapsulation

In Java 9 Module System, a Module is a First class citizen.
In Simple Terms, Java has these many First class citizens:

- In OOP (Object Oriented Programming) = Object
- In FP (Functional Programming) = Function or Lambda
- In Module System (Modular Programming) = Module

Java 9 Module Basics

We should remember the following important points about Java 9 Module:

- Each module has a unique Name.
- Each module has some description in a source file.
- A Module description is expressed in a source file called “module-info.java”.
- As “module-info.java” file describes a Module, it is also known as “Module Descriptor”.
- A Module Descriptor is a Java file. It is not an XML, Text file or Properties file.
- By convention, we should use same name “module-info.java” for Module Descriptor.
- By convention, Module Descriptor file is placed in the top level directory of a Module.
- Each Module can have any number of Packages and Types.
- As of now, year 2017, JDK 9 have 97 modules.
- We can create our own modules.
- One Module can dependent on any number of modules.
- Each Module should have one and only one Module Descriptor (“module-info.java”).

Using Java SE 9, we develop Modules that means we can do Modular Programming in Java.

3.1 JDK 9 and 11+ Modules

Java 9 Modules:

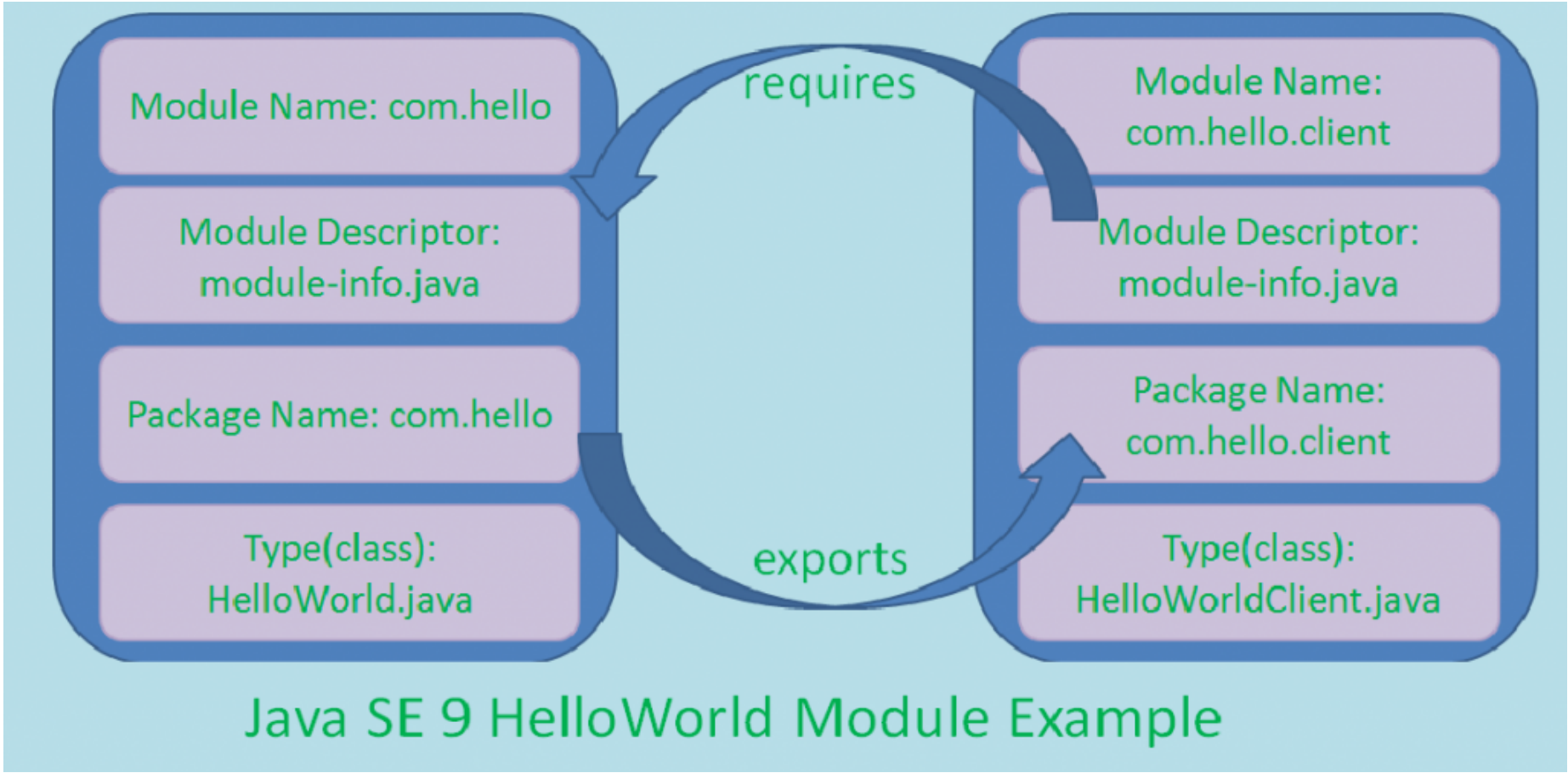
Steps to Develop a Java 9 Module

Follow these steps one by one to develop and test the “HelloWorld” Module.

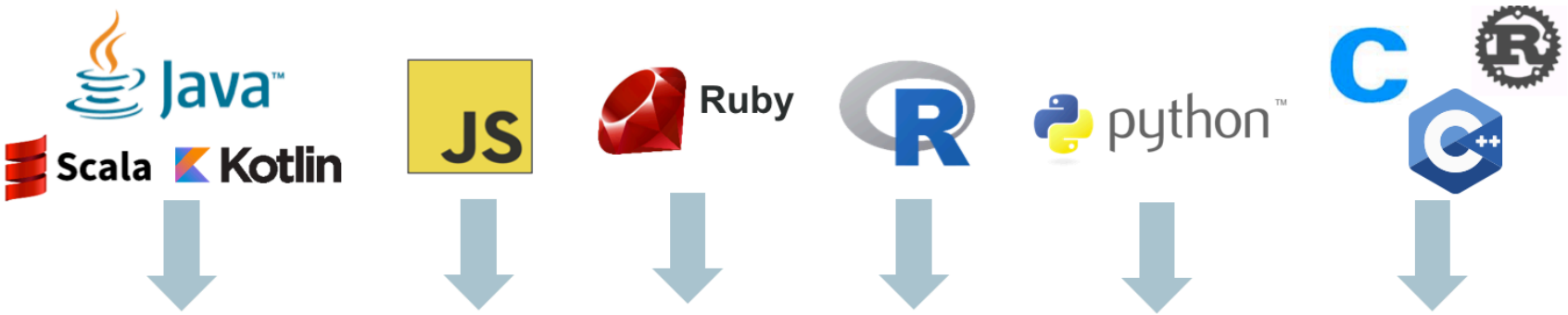
- Create Module name folder, for example “com.hello”.
- Create Module Packages, for example “com.hello”.
- Create our Java component, for example “HelloWorld.java”.
- Create Module Descriptor, for example “module-info.java”.
- Define Module Description in Module Descriptor, for example “exports com.hello;” of “module-info.java” in “com.hello” module.
- Create Module Jars if required.
- Test the modules.
- These steps are common for almost all modules development.

3.1 JDK 9 and 11+ Modules

Java 9 Modules:



3.1 GraalVM



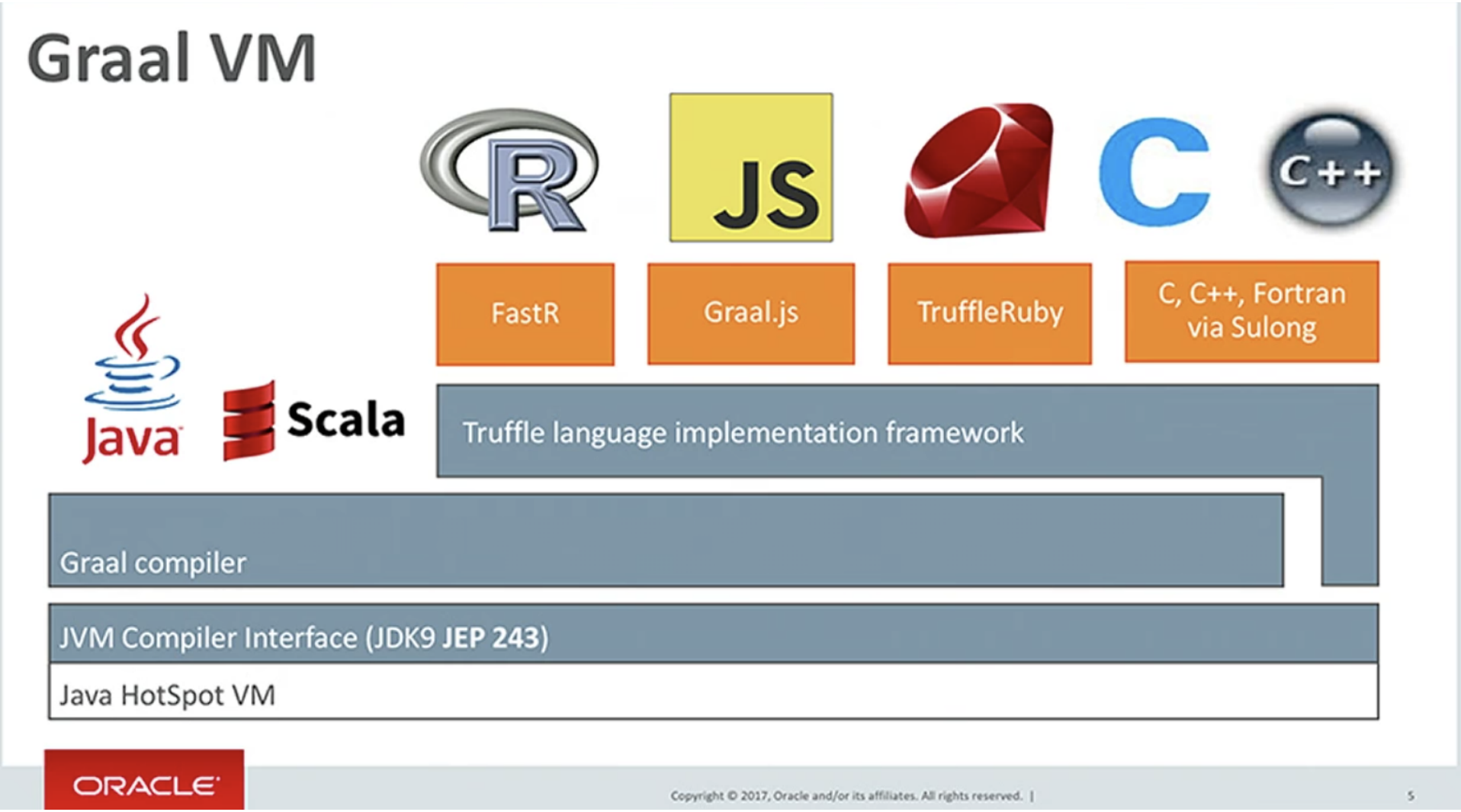
Automatic transformation of interpreters to compiler

GraalVM™

Embeddable in native or managed applications

GraalVM is an extension of the Java virtual machine to support more languages and execution modes. The Graal project includes a new high performance Java compiler, itself called Graal, which can be used in a just-in-time configuration on the HotSpot VM, or in an ahead-of-time configuration on the SubstrateVM.

3.1 GraalVM

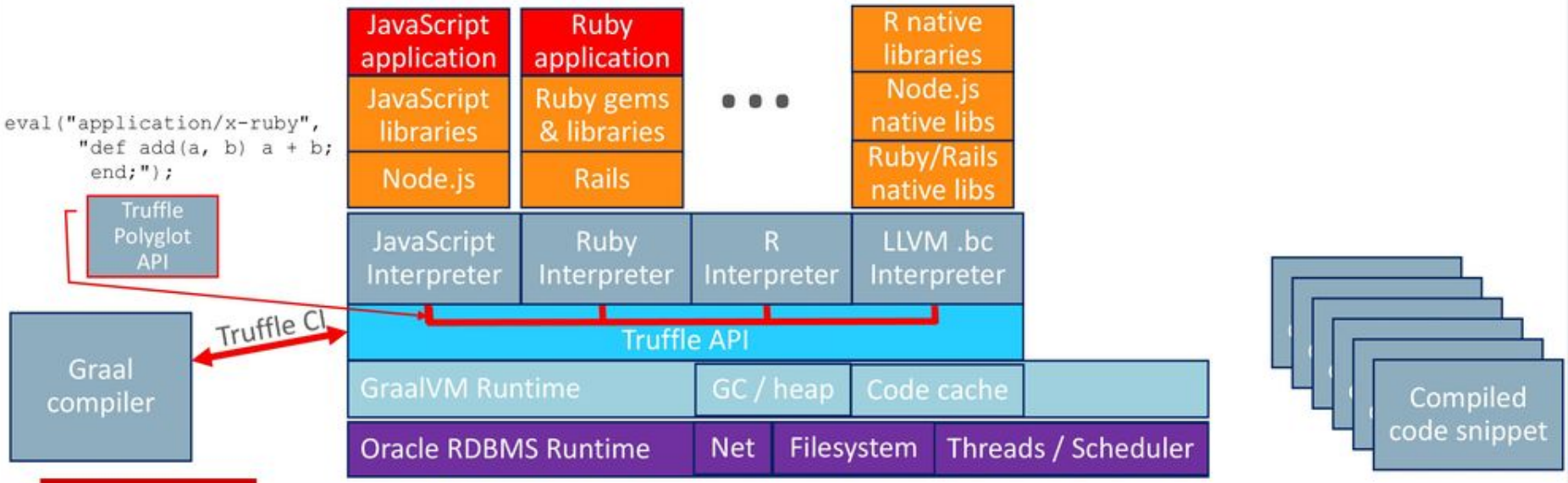


Copyright: https://www.graalvm.org/

3.1 GraalVM

GraalVM embedded in the Oracle Database

Embedding GraalVM allows substitution of the underlying engine's service layer



3.2 Build Automation: Apache ANT

Optional steps to build the lectures with Gradle – all lectures are independent by any Build
Automation system (e.g. Gradle, MVN, ANT, etc.) and the labs are projects in Eclipse IDE:

Apache ANT is a Java based build tool from Apache Software Foundation. Apache Ant's build files are written in XML and they take advantage of being open standard, portable and easy to understand.

Features:

- Ant is the most complete Java build and deployment tool available.
- Ant is platform neutral and can handle platform specific properties such as file separators.
- Ant can be used to perform platform specific tasks such as modifying the modified time of a file using 'touch' command.
- Ant scripts are written using plain XML. If you are already familiar with XML, you can learn Ant pretty quickly.
- Ant is good at automating complicated repetitive tasks.
- Ant comes with a big list of predefined tasks.
- Ant provides an interface to develop custom tasks.
- Ant can be easily invoked from the command line and it can integrate with free and commercial IDEs.

3.2 Build Automation: Apache ANT

```
<project>

    <target name="clean">
        <delete dir="build"/>
    </target>

    <target name="compile">
        <mkdir dir="build/classes"/>
        <javac srcdir="src" destdir="build/classes"/>
    </target>

    <target name="jar">
        <mkdir dir="build/jar"/>
        <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
            <manifest>
                <attribute name="Main-Class" value="oata.HelloWorld"/>
            </manifest>
        </jar>
    </target>

    <target name="run">
        <java jar="build/jar/HelloWorld.jar" fork="true"/>
    </target>

</project>
```

ant compile jar run

3.2 Build Automation: Apache ANT

While having a look at the buildfile, we will see some similar steps between Ant and the java-only commands:

java-only	Ant
`md build\classes`	`<mkdir dir="build/classes"/>`
`javac`	`<javac`
`-sourcepath src`	`srcdir="src"`
`-d build\classes`	`destdir="build/classes"/>`
`src\oata\HelloWorld.java`	`<!-- automatically detected -->`
`echo Main-Class: oata.HelloWorld>mf`	`<!-- obsolete; done via manifest tag -->`
`md build\jar`	`<mkdir dir="build/jar"/>`
`jar cfm`	`<jar`
`build\jar\HelloWorld.jar`	`destfile="build/jar/HelloWorld.jar"`
`mf`	
`-C build\classes`	`basedir="build/classes">`
`.`	`<manifest>`
	`<attribute name="Main-Class" value="oata.HelloWorld"/>`
	`</manifest>`
	`</jar>`
`java -jar build\jar\HelloWorld.jar`	`<java jar="build/jar/HelloWorld.jar" fork="true"/>`

ant compile

ant jar

ant run

3.2 Build Automation: Apache MVN - Maven

Optional steps to build the lectures with Gradle – all lectures are independent by any Build Automation system (e.g. Gradle, MVN, ANT, etc.) and the labs are projects in Eclipse IDE:

Apache Maven (MVN) is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. Using maven we can build and manage any Java based project.

Features:

- Maven uses Convention over Configuration, which means developers are not required to create build process themselves. Developers do not have to mention each and every configuration detail. Maven provides sensible default behavior for projects. When a Maven project is created, Maven creates default project structure. Developer is only required to place files accordingly and he/she need not to define any configuration in pom.xml.
- Simple project setup that follows best practices.
- Consistent usage across all projects.
- Dependency management including automatic updating.

3.2 Build Automation: Apache MVN - Maven

Apache Maven (MVN) Features:

- A large and growing repository of libraries.
- Extensible, with the ability to easily write plugins in Java or scripting languages.
- Instant access to new features with little or no extra configuration.
- **Model-based builds** – Maven is able to build any number of projects into predefined output types such as jar, war, metadata.
- **Coherent site of project information** – Using the same metadata as per the build process, maven is able to generate a website and a PDF including complete documentation.
- **Release management and distribution publication** – Without additional configuration, maven will integrate with your source control system such as CVS and manages the release of a project.
- **Backward Compatibility** – You can easily port the multiple modules of a project into Maven 3 from older versions of Maven. It can support the older versions also.
- **Automatic parent versioning** – No need to specify the parent in the sub module for maintenance.
- **Parallel builds** – It analyzes the project dependency graph and enables you to build schedule modules in parallel. Using this, you can achieve the performance improvements of 20-50%.
- **Better Error and Integrity Reporting** – Maven improved error reporting, and it provides you with a link to the Maven wiki page where you will get full description of the error.

3.2 Build Automation: Apache MVN - Maven

Creating a Project

You will need somewhere for your project to reside, create a directory somewhere and start a shell in that directory. On your command line, execute the following Maven goal:

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

If you have just installed Maven, it may take a while on the first run. This is because Maven is downloading the most recent artifacts (plugin jars and other files) into your local repository. You ma
execute the command a couple of times before it succeeds. This is because the remote server may time out before your downloads are complete. Don't worry, there are ways to fix that.

You will notice that the *generate* goal created a directory with the same name given as the artifactId. Change into that directory.

```
cd my-app
```

Under this directory you will notice the following standard project structure.

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

The `src/main/java` directory contains the project source code, the `src/test/java` directory contains the test source, and the `pom.xml` file is the project's Project Object Model, or POM.

3.2 Build Automation: Apache MVN - Maven

The POM

The pom.xml file is the core of a project's configuration in Maven. It is a single configuration file that contains the majority of information required to build a project in just the way you want. The POM is huge and can be daunting in its complexity, but it is not necessary to understand all of the intricacies just yet to use it effectively. This project's POM is:

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2.   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3.   <modelVersion>4.0.0</modelVersion>
4.
5.   <groupId>com.mycompany.app</groupId>
6.   <artifactId>my-app</artifactId>
7.   <version>1.0-SNAPSHOT</version>
8.   <packaging>jar</packaging>
9.
10.  <name>Maven Quick Start Archetype</name>
11.  <url>http://maven.apache.org</url>
12.
13.  <dependencies>
14.    <dependency>
15.      <groupId>junit</groupId>
16.      <artifactId>junit</artifactId>
17.      <version>4.8.2</version>
18.      <scope>test</scope>
19.    </dependency>
20.  </dependencies>
21. </project>
```

```
mvn clean build
mvn package
java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App
```

3.3 Build Automation: Gradle

Optional steps to build the lectures with Gradle – all lectures are independent by any Build Automation system (e.g. Gradle, MVN, ANT, etc.) and the labs are projects in Eclipse IDE:

Gradle is an advanced general purpose **build management system** based on Groovy.

Features:

- It supports automatic download and configuration of project dependencies (libraries).
- It supports Maven and Ivy (ANT) repositories for retrieving project dependencies, allowing reuse of the artifacts of existing build systems.
- It supports multi-project and multi-artifact builds.

(Optional full Gradle presentation @ISM – Cyber-Security Master – www.ism.ase.ro, by Marius Popa @ Oracle)

https://gradle.org/install/
https://plugins.gradle.org/
https://github.com/jabedhasan21/java-hello-world-with-gradle/blob/master/README.md
https://www.tutorialspoint.com/gradle/index.htm

3.3.1. Gradle Intro and Basics

Projects and tasks in **Gradle**:

- A Gradle build consists of one or more projects.
- A ***project*** using Gradle describes its build via a ***build.gradle*** file. ***build.gradle*** file is located in the root folder of the project. ***build.gradle*** file is based on a *Domain Specific Language* (DSL) written in Groovy.
- ***build.gradle*** build file defines a project and its tasks.
- A ***task*** represents a piece of work which a build performs.

```
task hello {
  doLast {
    println 'Hello Gradle'
    println 'Project name: ' + rootProject.name
  }
}
```

```
mepopa@MEPOPA-RO /d/Work/P006_IoT_GW/GradleTraining
$ gradle project
:projects

------------------------------------------------------------
Root project
------------------------------------------------------------

Root project 'GradleTraining'
No sub-projects

To see a list of the tasks of a project, run gradle <project-path>:tasks
For example, try running gradle :tasks

BUILD SUCCESSFUL

Total time: 3.676 secs
```

```
mepopa@MEPOPA-RO /d/Work/P006_IoT_GW/GradleTraining
$ gradle hello
:hello
Hello Gradle
Project name: GradleTraining

BUILD SUCCESSFUL

Total time: 3.162 secs
```

3.3.2. Installing and configuring Gradle

- ***Requirement***: JDK (Java Development Kit) installation.
- ***Installing***: Download the latest release of Gradle from http://gradle.org/gradle-download/ | https://gradle.org/install/ for the usage on the command line. Add the /bin folder to PATH environment variable.
- ***Gradle daemon***: avoid starting the Java virtual machine for every build. Usage: org.gradle.daemon=true in gradle.properties in the ${HOME}/.gradle; Executing gradle with the --daemon parameter on the command line or gradle --stop to stop it.
- Set specific JVM options: GRADLE_OPTS environment variable. Eg. export GRADLE_OPTS=-Xmx1024m defines 1 GB as maximum heap size.

3.3.3. Gradle plug-ins

- Plug-ins extend Gradle core functionality. Extension to Gradle adding some preconfigured tasks.
- Gradle ships with a number of plug-ins (java), and custom plug-ins can be developed.
- Adding a plug-in in ***build.gradle*** file: apply plugin: 'pluginname' (Eg. apply plugin: 'java').
- Registry for Gradle plug-ins: https://plugins.gradle.org/.

3.3.3. Gradle plug-ins

build.gradle

```
task hello {
  doLast {
    println 'Hello Gradle'
    println 'Project name: ' + rootProject.name
  }
}
```

```
mepopa@MEPOPA-RO /d/Work/P006_IoT_GW/GradleTraining
$ gradle tasks
:tasks

------------------------------------------------------------
All tasks runnable from root project
------------------------------------------------------------

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
wrapper - Generates Gradle wrapper files. [incubating]

Help tasks
----------
components - Displays the components produced by root project 'GradleTraining'. [incubating]
dependencies - Displays all dependencies declared in root project 'GradleTraining'.
dependencyInsight - Displays the insight into a specific dependency in root project 'GradleTraining'.
help - Displays a help message.
model - Displays the configuration model of root project 'GradleTraining'. [incubating]
projects - Displays the sub-projects of root project 'GradleTraining'.
properties - Displays the properties of root project 'GradleTraining'.
tasks - Displays the tasks runnable from root project 'GradleTraining'.

Other tasks
-----------
hello

To see all tasks and more detail, run gradle tasks --all

To see more detail about a task, run gradle help --task <task>

BUILD SUCCESSFUL

Total time: 3.418 secs
```

build.gradle

```
apply plugin: 'java'
task hello {
  doLast {
    println 'Hello Gradle'
    println 'Project name: ' + rootProject.name
  }
}
```

```
mepopa@MEPOPA-RO /d/Work/P006_IoT_GW/GradleTraining
$ gradle tasks
:tasks

------------------------------------------------------------
All tasks runnable from root project
------------------------------------------------------------

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles classes 'main'.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles classes 'test'.

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
wrapper - Generates Gradle wrapper files. [incubating]

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.
```

```
Help tasks
----------
components - Displays the components produced by root project 'GradleTraining'. [incubating]
dependencies - Displays all dependencies declared in root project 'GradleTraining'.
dependencyInsight - Displays the insight into a specific dependency in root project 'GradleTraining'.
help - Displays a help message.
model - Displays the configuration model of root project 'GradleTraining'. [incubating]
projects - Displays the sub-projects of root project 'GradleTraining'.
properties - Displays the properties of root project 'GradleTraining'.
tasks - Displays the tasks runnable from root project 'GradleTraining'.

Verification tasks
------------------
check - Runs all checks.
clean - Deletes the build directory.
test - Runs the unit tests.

Other tasks
-----------
hello

Rules
-----
Pattern: clean<TaskName>: Cleans the output files of a task.
Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.
Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.

To see all tasks and more detail, run gradle tasks --all

To see more detail about a task, run gradle help --task <task>

BUILD SUCCESSFUL

Total time: 5.03 secs
```

3.3.4. Gradle dependency management

- Managing the classpath of Gradle projects: adding JAR files, directories or other projects to the build path of the application.
- Automatic download of Java library dependencies, specifying the dependency in Gradle build file.
- A Java library is identified by Gradle via its project's groupId:artifactId:version (also known as GAV in Maven).
- Adding a dependency: new entry in the dependency section in ***build.gradle*** file.

```
repositories {
    jcenter()
}

dependencies {
    compile fileTree(dir: 'libs',
include: ['*.jar'])
    testCompile 'junit:junit:4.12'
}
```

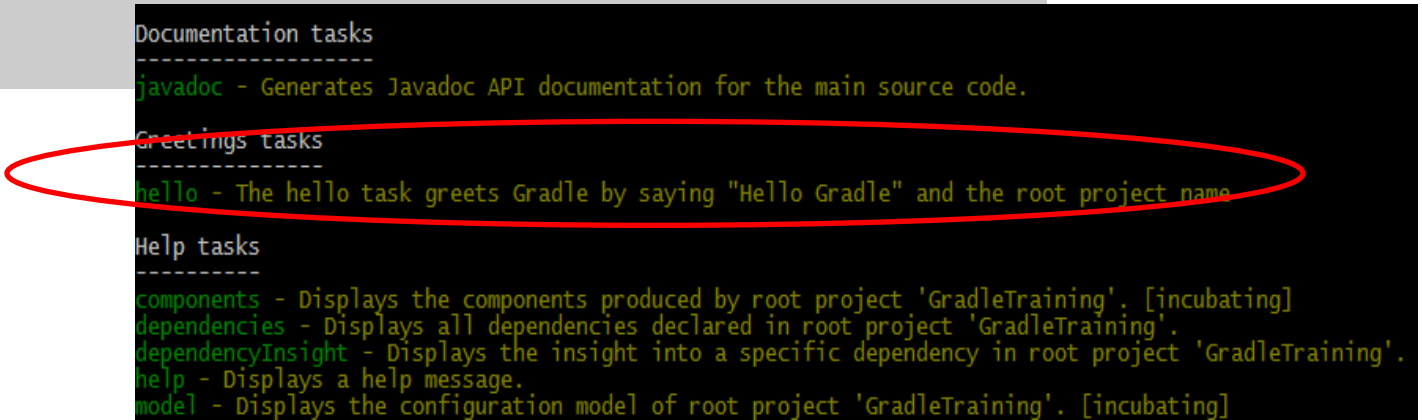
```
mepopa@MEPOPA-RO /d/Work/P006_IoT_GW/GradleTraining
$ gradle clean build
:clean UP-TO-DATE
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:jar
:assemble
:compileTestJava UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:check UP-TO-DATE
:build

BUILD SUCCESSFUL

Total time: 4.357 secs
```

3.3.5. Gradle tasks

- Default Gradle tasks: tasks for introspection of Gradle itself (eg. task tasks).

- Custom Gradle tasks: tasks created by developers (eg. task hello). Running the gradle tasks task, the hello task will be listed under ***Other tasks***. Tasks without a group are considered as private tasks. Groups can be applied with the group property and a description can be applied by using the description property .

```
task hello {

    group 'greetings'
    description 'The hello task greets Gradle by
saying "Hello Gradle" and the root project name'

    doLast {
        println 'Hello Gradle'
        println 'Project name: ' + rootProject.name
    }
}
```



3.3.6. Building Java projects

- Java plug-in: provides tasks to compile Java source code, run unit tests, create Javadoc and create a JAR file.
- Default project layout:

  - Java source code: src/main/java.

  - Java tests: src/test/java.

- Different project structure: sourceSets.

```
sourceSets {
  main {
     java {
        srcDir 'src'
        }
     }
  test {
     java {
        srcDir 'test'
        }
     }
}
```

- Start the build: gradle build for HelloWorld.java.

```
mepopa@MEPOPA-RS /d/Work/P006_IoT_GW/GradleTraining
$ gradle build
I am not invoked, but I always get printed
:compileJava
:processResources UP-TO-DATE
:classes
:jar
:assemble
:compileTestJava UP-TO-DATE
:processTestResources UP-TO-DATE
:testClasses UP-TO-DATE
:test UP-TO-DATE
:check UP-TO-DATE
:build

BUILD SUCCESSFUL

Total time: 5.434 secs
```

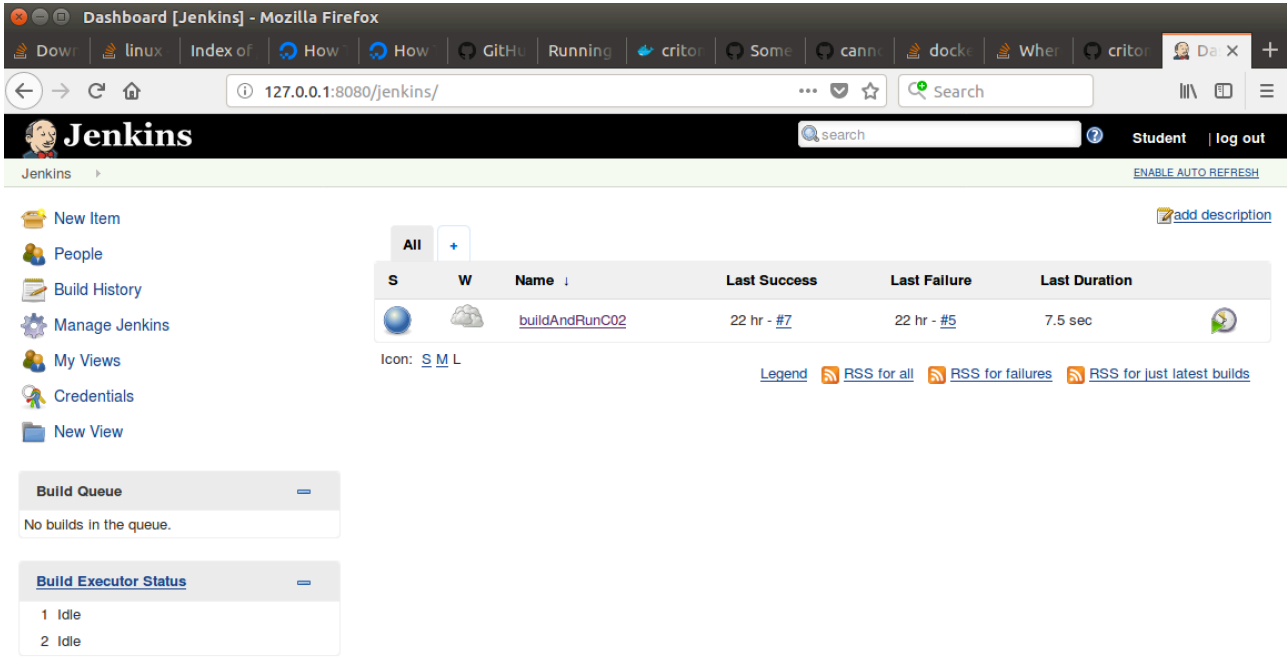
3.4 Jenkins/Hudson – DevOps Build Automation

```
export JAVA_HOME=/opt/software/java/jdks/jdk1.8.0_161
export CLASSPATH=.:$JAVA_HOME/jre/lib
export CATALINA_HOME=/opt/software/apache-tomcat-9.0.4
export
PATH=.:$JAVA_HOME/bin:$CATALINA_HOME/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games
```

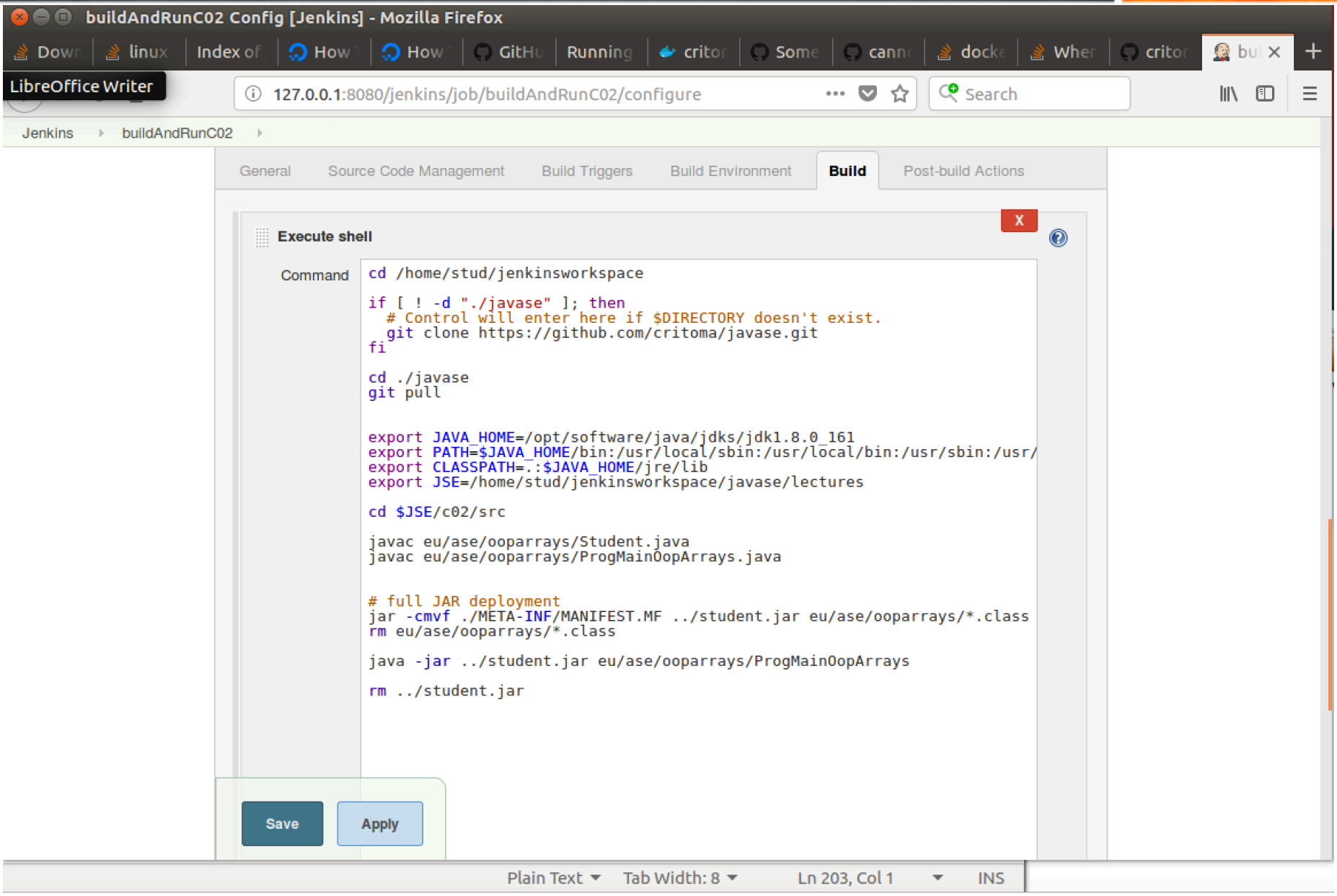
```
cd $CATALINA_HOME
bin/startup.sh
```

http://127.0.0.1:8080
http://127.0.0.1:8080/jenkins/

```
bin/shutdown.sh
```



3.4 Jenkins/Hudson – DevOps Build Automation



3.5 Deployment: Std. OS vs. VMs vs. Docker Containers

Docker image:
https://hub.docker.com | https://hub.docker.com/r/critoma/ubuntu-java-node-py-dev/

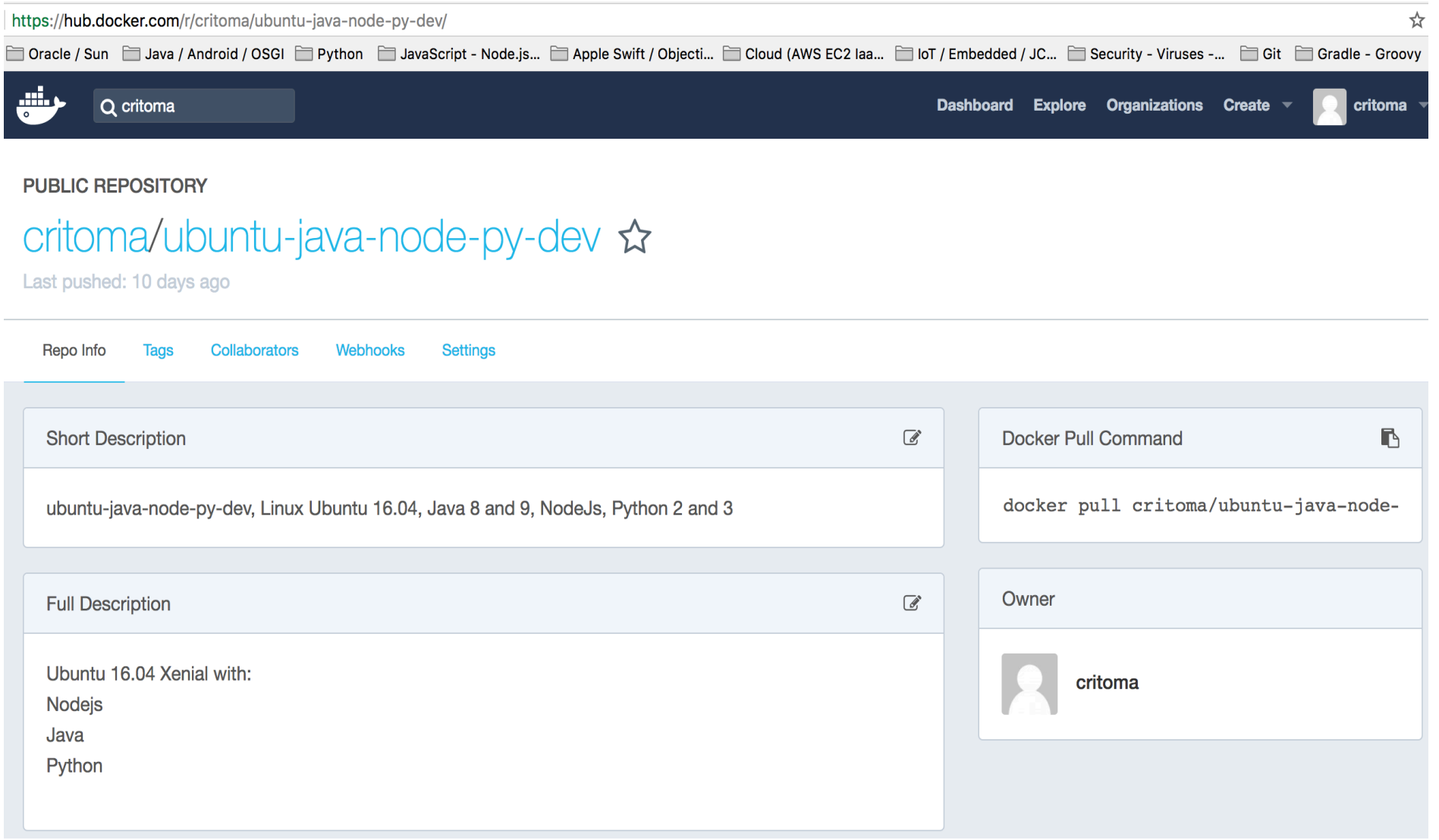
Install Docker in Linux (similar steps used for Windows):
https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-from-a-package
https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/
sudo dpkg -i /path/to/package.deb
sudo docker run hello-world
docker search critoma
docker pull critoma/ubuntu-java-node-py-dev
docker run -it critoma/ubuntu-java-node-py-dev

Resources:
https://www.tutorialspoint.com/docker/docker_architecture.htm
https://www.tutorialspoint.com/docker/docker_containers_and_shells.htm
https://docs.docker.com/install/linux/docker-ce/ubuntu/
https://docs.docker.com/engine/installation/linux/linux-postinstall/
https://docs.docker.com/get-started/
https://docs.docker.com/get-started/part2/
https://docs.docker.com/get-started/part6/

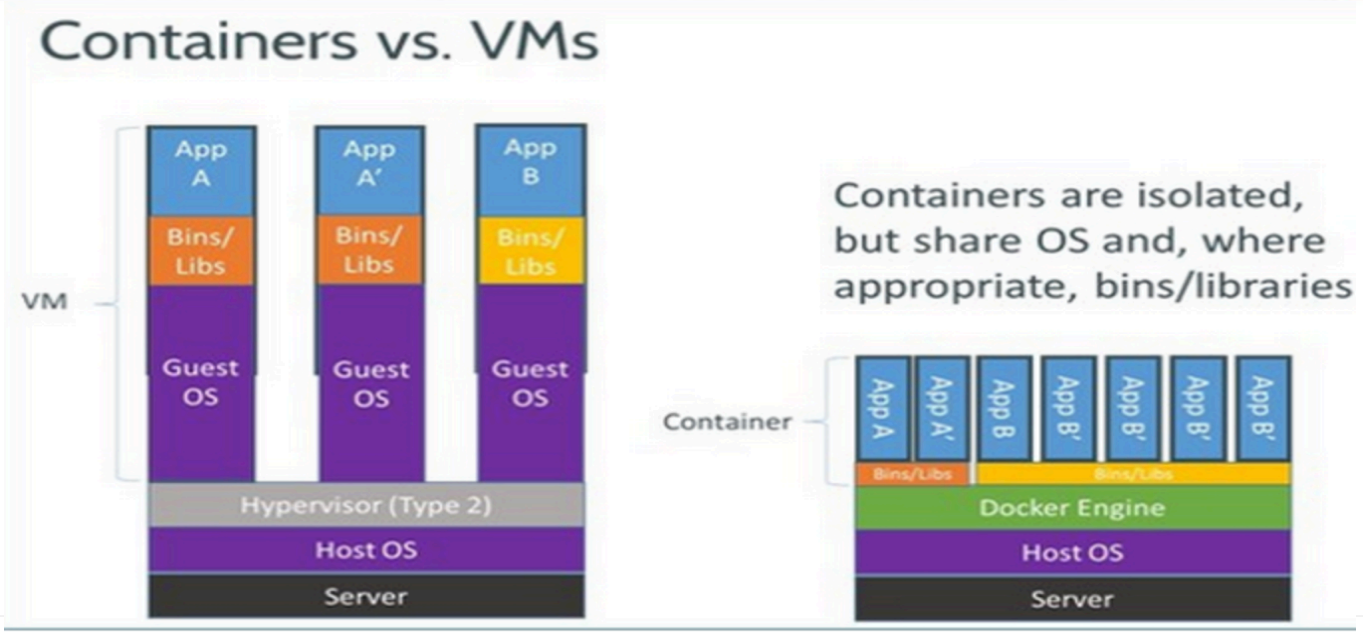
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04
https://www.digitalocean.com/community/tutorials/how-to-remove-docker-images-containers-and-volumes#purging-all-unused-or-dangling-images,-containers,-volumes,-and-networks
https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-16-04
http://fabiorehm.com/blog/2014/09/11/running-gui-apps-with-docker/
https://github.com/fgrehm/docker-eclipse
http://fabiorehm.com/blog/2014/09/11/running-gui-apps-with-docker/
https://stackoverflow.com/questions/19234831/where-are-docker-images-stored-on-the-host-machine

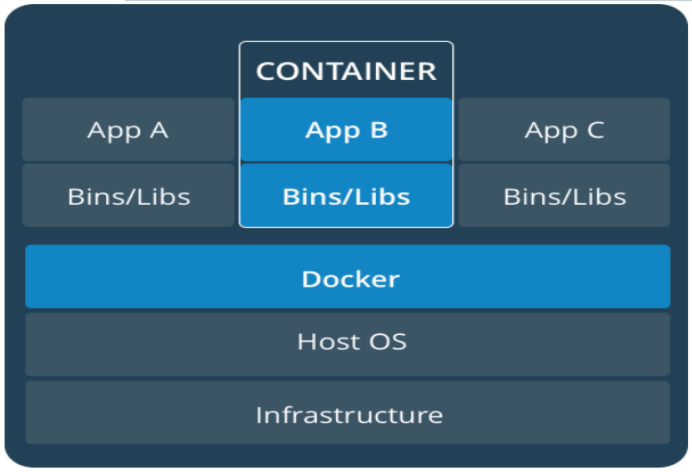
3.5 Deployment: Std. OS vs. VMs vs. Docker Containers

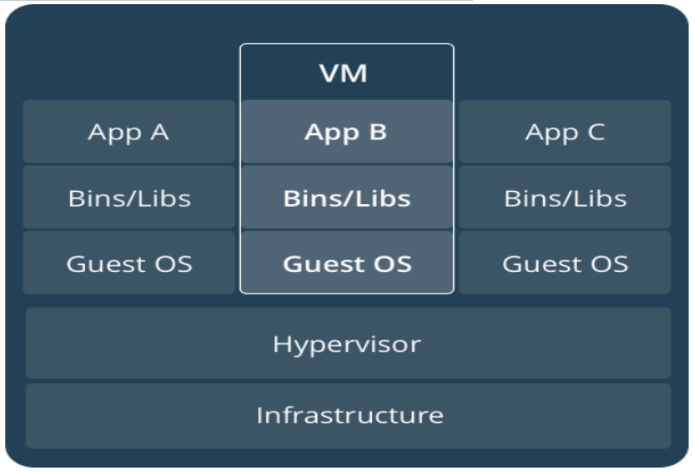
https://hub.docker.com/r/critoma/ubuntu-java-node-py-dev/



3.5 Deployment: Std. OS vs. VMs vs. Docker Containers







Download Oracle VM VirtualBox OVA:
https://drive.google.com/drive/folders/1SI7uzNPbQ3YsFo3_sbkE3R5lf2f1psZO?usp=sharing
Download Docker Container: https://hub.docker.com/r/critoma/ubuntu-java-node-py-dev/

Questions & Answers!

Thanks!

Java Application Development
End of Lecture 2 – summary of Java SE