ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

OBJECT-ORIENTED PROGRAMMING

Array - String

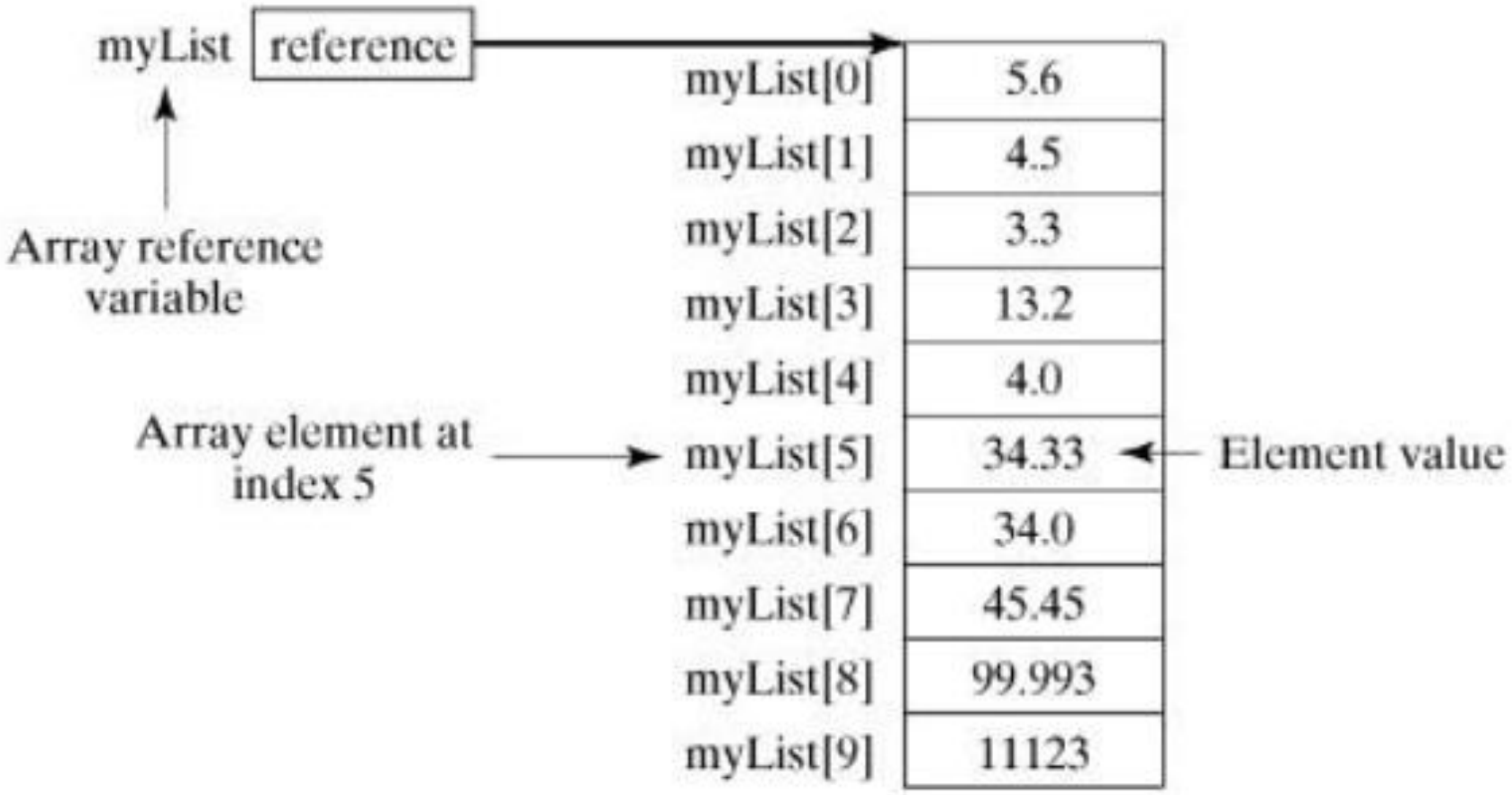
Outline

- Array:
  - Create array
  - Process array
  - Foreach loops
  - Passing Arrays to Methods
  - Return Arrays from Methods
  - Arrays class

Outline

- String:
  - charSequence Interface
  - Create string
  - String methods

Arrays


myList
reference
Array reference variable
Array element at index 5
myList[0]
myList[1]
myList[2]
myList[3]
myList[4]
myList[5]
myList[6]
myList[7]
myList[8]
myList[9]
5.6
4.5
3.3
13.2
4.0
34.33
34.0
45.45
99.993
11123
Element value

Arrays

- Create Arrays

dataType[] arrayRefVar = new dataType[arraySize];

- Example:

double[] myList = new double[10];

Process Arrays

We often use either **for** loop or **foreach** loop because all of the elements in an array are of the same type and the size of the array is known

```
public class TestArray {

   public static void main(String[] args) {
      double[] myList = {1.9, 2.9, 3.4, 3.5};

      // Print all the array elements
      for (int i = 0; i < myList.length; i++) {
         System.out.println(myList[i] + " ");
      }

      // Summing all elements
      double total = 0;
      for (int i = 0; i < myList.length; i++) {
         total += myList[i];
      }
      System.out.println("Total is " + total);

      // Finding the largest element
      double max = myList[0];
      for (int i = 1; i < myList.length; i++) {
         if (myList[i] > max) max = myList[i];
      }
      System.out.println("Max is " + max);
   }
```

Output

```
1.9
2.9
3.4
3.5
Total is 11.7
Max is 3.5
```

Foreach Loops

- JDK 1.5 introduced a new for loop known as foreach loop or enhanced for loop, which enables you to traverse the complete array sequentially without using an index variable

```
public class TestArray {

   public static void main(String[] args) {
      double[] myList = {1.9, 2.9, 3.4, 3.5};

      // Print all the array elements
      for (double element: myList) {
         System.out.println(element);
      }
   }
}
```

Output

```
1.9
2.9
3.4
3.5
```

Passing Arrays to Methods

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

- We can invoke it by passing an array. For example, the following statement invokes the printArray method to display 3, 1, 2, 6, 4, and 2

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

Return an Array from a Method

- A method may also return an array. For example, the following method returns an array that is the reversal of another 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;
}
```

Java String

- charSequence Interface
- Create string
- String methods

Java String

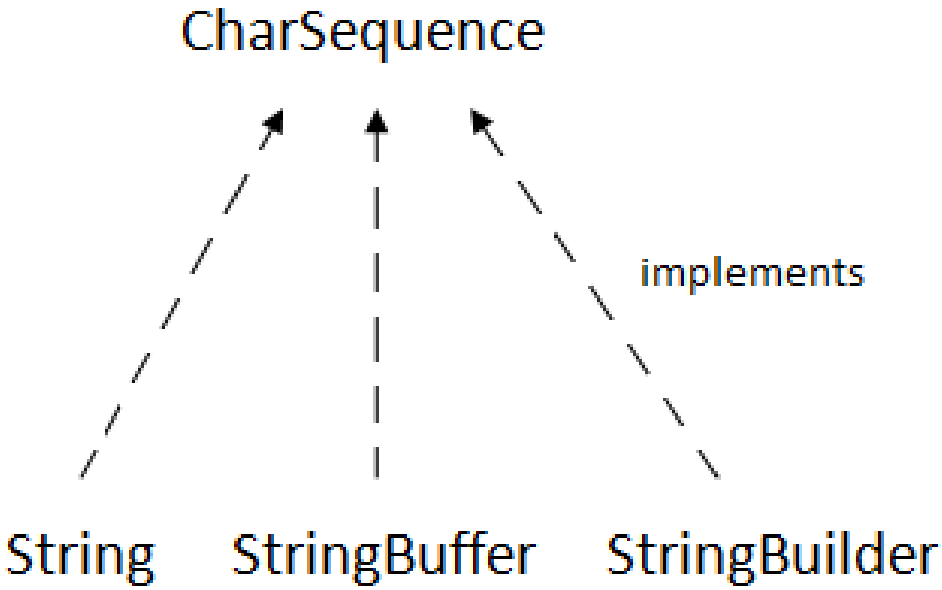
- In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
String s=new String(ch);
```

is same as:

```
String s="javatpoint";
```

CharSequence Interface



- The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created. For mutable strings, you can use StringBuffer and StringBuilder classes (

Create String

- There are two ways to create String object:
  - By string literal
  - By new keyword

Create String

- By string literal

  String s="welcome";

- Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

```
String s1="Welcome";
String s2="Welcome";//It doesn't create a new instance
```

Create String

- By new keyword

String s=**new** String("Welcome");//creates two objects and one reference variable

- In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool)

Create String

```
public class StringExample{
public static void main(String args[]){
String s1="java";//creating string by java string literal
char ch[]={'s','t','r','i','n','g','s'};
String s2=new String(ch);//converting char array to string
String s3=new String("example");//creating java string by new keyword
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

```
java
strings
example
```

Create String

```
public class StringExample{
public static void main(String args[]){
String s1="java";//creating string by java string literal
char ch[]={'s','t','r','i','n','g','s'};
String s2=new String(ch);//converting char array to string
String s3=new String("example");//creating java string by new keyword
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

```
java
strings
example
```

String methods

No.	Method	Description
1	char charAt(int index)	returns char value for the particular index
2	int length()	returns string length
3	static String format(String format, Object... args)	returns a formatted string.
4	static String format(Locale l, String format, Object... args)	returns formatted string with given locale.
5	String substring(int beginIndex)	returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	returns true or false after matching the sequence of char value.
8	static String join(CharSequence delimiter, CharSequence... elements)	returns a joined string.
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	returns a joined string.
10	boolean equals(Object another)	checks the equality of string with the given

String methods

11	boolean isEmpty()	checks if string is empty.
12	String concat(String str)	concatenates the specified string.
13	String replace(char old, char new)	replaces all occurrences of the specified char value.
14	String replace(CharSequence old, CharSequence new)	replaces all occurrences of the specified CharSequence.
15	static String equalsIgnoreCase(String another)	compares another string. It doesn't check case.
16	String[] split(String regex)	returns a split string matching regex.
17	String[] split(String regex, int limit)	returns a split string matching regex and limit.
18	String intern()	returns an interned string.
19	int indexOf(int ch)	returns the specified char value index.
20	int indexOf(int ch, int fromIndex)	returns the specified char value index starting with given index.
21	int indexOf(String substring)	returns the specified substring index.

String methods

22	int indexOf(String substring, int fromIndex)	returns the specified substring index starting with given index.
23	String toLowerCase()	returns a string in lowercase.
24	String toLowerCase(Locale l)	returns a string in lowercase using specified locale.
25	String toUpperCase()	returns a string in uppercase.
26	String toUpperCase(Locale l)	returns a string in uppercase using specified locale.
27	String trim()	removes beginning and ending spaces of this string.
28	static String valueOf(int value)	converts given type into string. It is an overloaded method.

Read more

- StringBufer class: https://www.javatpoint.com/StringBuffer-class
- StringBuilder class: https://www.javatpoint.com/StringBuilder-class

Exercises

Arrays

1. Input and Output a 1D-Array with n integer values
2. List negative values in the array
3. Find max value in the array
4. Find the first position of negative values in the array
5. Caculate sum of values in the array
6. Sort the array in ascending order
7. Add an element x to the first postion of the aray
8. Delete the first greater 0 element in the array

Exercises

String:

Given a Vietnamese fullname (ex “Nguyen Van Teo”). Students need to write methods as follows:

1. Count how many words in the name.
2. Return a first name
3. Return a last name
4. Return a middle name
5. Capitalize the first character in each word of the name
6. Formalize the name, including:
   - Delete all spaces in front and behind of the name.
   - Leave one space between the words of the name.

End of file