Java

Note for learning the language

Fahad Pathan

14.09.2021
fahadpathan56@gmail.com

Run on cmd:

one file

javac fileName.java
java fileName.java

whole project

javac -d . *.java
java packageName.fileNameWithMainFunction

Data types:

primitive

int, byte, short, long, boolean, char, float, double

wrapper class

Integer, Byte, Short, Long, Boolean, Character, Float, Double

Input:

java.util.Scanner; // class to use Scanner

taking user input

```
Scanner sc = new Scanner(System.in); // Scanner is used to take input from user
int a;
a = sc.nextInt(); // taking integer input

double d = sc.nextDouble(); // taking double value
float f = sc.nextFloat(); // taking float value
long l = sc.nextLong(); // taking long value

String s = sc.next(); // to take string
String str = sc.nextLine(); // to take a line as input
char ch = sc.next().charAt(0); // to take one character as input
```

Array:

tutorials:

https://www.tutorialspoint.com/java/java_arrays.htm

array declaration

```
int[] ar; // preferred way
int ar[]; // works but not preferred
```

initialization

```
ar = new int[10]; // here 10 is the size of the array.

// using one line to declare and create.
int[] ar = new int[10];
int[] ar = {1, 2, 3, 4, 5};
```

creation and declaration of 2D array

```
int[] ar1 = new int[4][5];
```

array length

```
int len = ar.length; // to get the length of array. (No parenthesis)
```

sort an array (ascending order)

```
Arrays.sort(ar);
```

searching in an array
```
// elements must be sorted in ascending order in the array. if the value is not found it will return
// negative value.
Arrays.binarySearch(ar, 8); // here searching 8 in the array
```

compare two array
```
// it will return true if all the elements of the arrays completely match (every index of elements
// of both array should match). otherwise it will return false.
Boolean b = Arrays.equals(ar1, ar2);
```

converting whole array into a string
```
String arrayString = Arrays.toString(ar);
System.out.println(arrayString); // can be printed like this. but if we print the array directly it will
        // print reference value only not the array elements.
```

ArrayList:
```
java.util.ArrayList; // class to use ArrayList
```

declaration and initialization
```
ArrayList<Integer> arList = new ArrayList<Integer>();
```

size
```
arList.size();
```

adding elements
```
arList.add(2);
arList.add(3, 45); // (index, element) to set location and add element. it will only possible if the
        // index already exists. we will add 45 on index 3. and the previous element of index 3 will
        // be shifted to index 4.  index 4 will be shifted to 5. and so on
```

replacing element
```
arList.set(3, 40); // (index, element) it will replace previously stored element in that index.
```

getting an element
```
int x = arList.get(2); // (index) it will return the element of index 2.
```

removing element
```
arList.remove(2); // (index) it will remove the element of index 2
```

clear whole ArrayList
```
arLIst.clear();
arList.removeAll(arList); // another way
```

check empty ArrayList
```
boolean check = arList.isEmpty(); // will ruturn true if empty otherwise false.
```

checking presence of an element
```
boolean present = arList.contains(30); // true, if present. otherwise false.
```

searching for an element
```
int pos = arList.indexOf(40); // for first occurrence. if element exists it will return index otherwise
        // will return -1.
int pos = arList.lastIndexOf(40); // for last occurrence
```

size of ArrayLlist
```
int size = arList.size();
```

copying an ArrayList to another
```
arList1.addAll(arList); // it will copy (append to arList1) all elements of arList to arList1.
```

cloning an ArrayList to another
```
arList1.clone(arList); // it will clone arList to arList1
```

equality check of two ArrayList
```
boolean equal = arList.equals(arList1);
```

sorting ArrayList
```
Collections.sort(arList); // ascending order
Collections.sort(arList, Collections.reverseOrder()); // descending order
```

printing an ArrayList
```
// directly
System.out.println(arList);

// using for-each loop
for (int x : arList) {
        System.out.println(x);
}

// using iterator
Iterator it = arList.iterator();
while(it.hasNext()) {
        System.out.println(it.next());
}
```

String:
declaration and initialization
```
String str = “Fahad Pathan”;
String str = new String(“Fahad Pathan”); // another way
```

length
```
int len = str.length();
```

equality of two string
```
boolen equal = str1.equals(str2);

// with ignoring case
boolean equal = str1.equalsIgnoreCase(str2);
```

checking presence of an element
```
boolean con = str.contains(“han”);
```

empty check of a string
```
boolean b = str.isEmpty();
```

concat of two string
```
str3 = str1 + str2;
str3 = str1.concat(str2); // another way
```

convert to uppercase and lowercase
```
str1 = str.toUpperCase();
str1 = str.toLowerCase();
```

checking the start and end
```
boolean b = str.startsWith("F");
boolean b = str.endsWith("abc");
```

getting any character of a string
```
char ch = str.charAt(3); // (index) will return the character of that index. if we use int instead of char
        // (int ch) it will return ascii value.
```

getting ascii value of any character of a string
```
int val = str.codePointAt(4); // (index) will return the ascii value of the element of index 4
```

searching for an element
```
int pos = str.indexOf("ha"); // first occurrence
int pos = str.indexOf("ha"); // last occurrence

// searching starts from a given index
str.indexOf("ha, 8); // it will start searching from index 8
```

removing spaces from start and end
```
String str1 = str.trim();
```

replacing any element
```
String str1 = str.replace('l', 'j'); // (old, new) it will replace all l with j
```

splitting a string
```
String[] str1 = str.split(" "); // it will split str where it finds a space.
```

substring
```
String str1 = str.substring(2); // (from) sub string from index 2 to the end of str.
String str1 = str.substring(4, 7); // (from, before that) sub string from index 4 to 6 (before 7)
```

converting string to another data type
```
String s = "100";
int i = Integer.parseInt(str);
int I = Integer.valueOf(str); // another way
```

converting any data type to String
```
int i = 100;
String str = Integer.toString(i);
```

StringBuffer
```
// The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be
// changed.
```

tutorial:
https://www.javatpoint.com/StringBuffer-class

declaration and initialization
```
StringBuffer sb = new StringBuffer();
```

inserting elements
```
// can append anything (int, float, String, boolean)
```

```
sb.append(1);
sb.append("abc");
sb.append(4.55);
sb.append(true);

// appending one StringBuffer to another
sb1.append(sb);
```

deleting elements
```
sb.delete(1, 3); // (from, before that) here from index 1 to 2 (before 3)
```

index of any element
```
int idx = sb.indexOf("abc");
```

element of an index
```
char c = sb.charAt(10); // it will return character of index 10. (will return ascii value if use int)
```

substring
```
String str = sb.substring(1, 3); // (from, before that) here from index 1 to 2 (before 3)
```

replacing
```
sb.replace(1, 5, "fahad"); // (from, before that, new element). // it will replace the elements from
        // index 1 to 4 with fahad
```

reversing
```
sb.reverse();
```

length
```
int len = sb.length();
```

fixing length
```
sb.setLength(5);
```

comparing two StringBuffer
```
int c = sb.compareTo(sb1); // it will return length difference. 0 if equal. positive if sb > sb1.
         // negative if sb < sb1
```

StringBuilder:
(functions are same as StringBuffer)

tutorial:
https://www.javatpoint.com/StringBuilder-class
https://www.geeksforgeeks.org/stringbuilder-class-in-java-with-examples/

difference between StringBuffer and StringBuilder
https://www.javatpoint.com/difference-between-stringbuffer-and-stringbuilder

Decimal, Binary, Ocatal, Hexadecimal:
conversion from decimal
```
int decimal = 15;
String binary = Integer.toBinaryString(decimal); // to binary
String octal = Integer.toOctalString(decimal); // to octal
String hexa = Integer.toHexString(decimal); // to hexadecimal
```

conversion to decimal

```
String binary = “1010”;
Integer decimal = Integer.parseInt(binary, 2); // (string, base)

String oct = “16”;
Integer decimal = Integer.parseInt(oct, 8);

String hex = “A”;
Integer decimal = Integer.parseInt(hex, 16);
```

Random:

declaration and initialization

```
Random rand = new Random();

int r = rand.nextInt(10) + 2; // (range) + fromWhere; it will return random value from 2 to 11.
```

Variable length argument:

```
// a function which recieves unlimited number of parameters

void add(int … num) {
	int sum = 0;
	for (int x : num) {
		sum += x;
	}
	System.out.println(sum);
}

add(2, 10); // valid call
add(3, 6, -1, 5); // valid call
add(4, 10, 5); // valid call
```

Exception handling:

tutorials

https://www.javatpoint.com/exception-handling-in-java
https://rollbar.com/guides/java/how-to-handle-exceptions-in-java/#

keywords

try, catch, finally, throw, throws

try-catch-finally block

```
// if no exception occurs try will work, if exception occurs catch will work, finally will always work.

try {
	int x = 10, y = 0;
	int r = x / y;
	System.out.println(r);

} catch (Exception e) {
	System.out.println(“Exception : “ + e);

} finally {
	… … … // something
}
```

File:

creating directory

```
File dir = new File("E:/Java"); // it will set the location of the directory

dir.mkdir(); // it will create the directory named Java in E drive
```

creating file

```
File f1 = new File("E:/Java/file.txt"); // it will set the location of file

// it is a must to use try-catch block to create file
try {
	f1.createNewFile(); // it will create the file on that location

} catch (Exception e) {
	System.out.println(e);
}
```

writing on a file

```
// it is a must to use try-catch block to write on a file
// there is another class named Formatter for writing on file. but I prefer FileWriter

try {
	FileWriter f = new FileWriter("E:/Java/file.txt"); // location of the file
	f.write("I am learning about file"); // writing
	f.newLine(); // for a new line

	f.close(); // must close the file after writing. otherwise nothing will add.

} catch (Exception e) {
	System.out.println("Something went wrong");
}
```

buffered writing

```
// it will append new writings on the file. (it will not erase previous writings).
// must use try-catch block like previous one.

FileWriter file = new FileWriter("E:/Java/file.txt");
BufferedWriter b = new BufferedWriter(file);

b.write("something");

b.close();
file.close();
```

reading a file

```
try {
	File f = new File("E:/Java/file.txt");
	Scanner sc = new Scanner(f);

	while(sc.hasNext()) {
		String str = sc.next();
		System.out.println(str);
	}

	sc.close(); // must close scanner
```

```
} catch (Exception e) {
        System.out.println(e);
}
```

some special functions

```
// getting the location
String loc = dir.getAbsolutePath();

// getting the name
String name = dir.getName();

// existence check
boolean b = dir.exists(); // will return false if exists

// deleting
dir.delete();
```

BigInteger

tutorial

https://www.geeksforgeeks.org/biginteger-class-in-java/

class

java.math.BigInteger

declaration and initialization

```
BigInteger A = new BigInteger("10000"); // string

String str = "5000"
BigInteger A = new BigInteger(str); // this is also valid

// another way
int a = 500;
BigInteger A = BigInteger.valueOf(a);
```

mathematical operations

addition

```
C = A.add(B); // c = a + b
```

subtraction

```
C = A.subtract(B); // c = a – b
```

multiplication

```
C = A.multiply(B); // c = a * b
```

dividation

```
C = A.divide(B); // c = a / b
```

remainder

```
C = A.remainder(B); // c = a % b
```

pow

```
int x = 5;
BigInteger B = A.pow(x); // x must be an integer
```

square root

```
B = A.sqrt(); // here both A and B are BigInteger
```

absolute value

```
B = A.abs();
```

extracting of value from BigInteger

```
int x   =  A.intValue();
long y   = A.longValue();
String str = A.toString();
```

comparison

```
int x = A.compareTo(B); // (return type is integer) compareTo returns -1 (less than), 0 (Equal),
        // 1 (greater than) according to values.
```

equality check

```
boolean b = A.equals(B); // return true if equals. otherwise false.
```