Core Java Cheat Sheet

**Java** is an open source programming language that has been changing the face of the IT market since ages. It is widely preferred by the programmers as the code written in Java can be executed securely on any platform, irrespective of the operating system or architecture of the device. The only requirement is, Java Runtime Environment (JRE) installed on the system.

­

Primitive Data Types

Let’s start off by learning the primitive [**data types**](https://www.edureka.co/blog/java-tutorial/#datatype) that Java offers:

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Range** |
| *byte* | 8 | -128..127 |
| *short* | 16 | -32,768..32,767 |
| *int* | 32 | -2,147,483,648.. 2,147,483,647 |
| *long* | 64 | -9,223,372,036,854,775,808.. 9,223,372,036,854,775,807 |
| *float* | 32 | 3.4e-0.38.. 3.4e+0.38 |
| *double* | 64 | 1.7e-308.. 1.7e+308 |
| *char* | 16 | Complete Unicode Character Set |
| *Boolean* | 1 | True, False |

Java Operators

There are mainly 8 different types of [**operators**](https://www.edureka.co/blog/java-tutorial/#operator) available in Java:

|  |  |
| --- | --- |
| **Operator Type** | **Operators** |
| Arithmetic | +, – , \*, ? , % |
| Assignment | =, +=, -=, \*=, /=, %=, &=, ^=, |=, <<=, >>=, >>>= |
| Bitwise | ^, &, | |
| Logical | &&, || |
| Relational | <, >, <=, >=,==, != |
| Shift | <<, >>, >>> |
| Ternary | ?: |
| Unary | ++x, –x, x++, x–, +x, –x, !, ~ |

Java Variables

[**Variables**](https://www.edureka.co/blog/java-tutorial/#variables) in Java refers to the name of the reserved memory area. You need variables to store any value for the computational or reference purpose.

 There are 3 types of variable in Java:

1. Local Variables
2. Instance Variables
3. Static Variables

{public | private} [static] type name [= expression | value];

Java Methods

A method is a set of code that is grouped together to perform a specific operation. A method is completed in two steps:

1. Method Initialization
2. Method Invocation

A method can be invoked either by calling it by reference or by value.

{public | private} [static] {type | void} name(arg1, ..., argN ){statements}

Data Conversion

The process of changing a value from one data type to another type is known as data type conversion. Data Type conversion is of two types:

1. *Widening:* The lower size datatype is converted into a higher size data type without loss of information.
2. *Narrowing:* The higher size datatype is converted into a lower size data type with a loss of information.

// Widening (byte<short<int<long<float<double)

int i = 10; //int--> long

long l = i; //automatic type conversion

// Narrowing

double d = 10.02;

long l = (long)d; //explicit type casting

// Numeric values to String

String str = String.valueOf(value);

// String to Numeric values

int i = Integer.parseInt(str);

double d = Double.parseDouble(str);

User Input

Java provides three ways to take an input from the user/ console:

1. Using BufferReader class
2. Using Scanner class
3. Using Console class

// Using BufferReader

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String name = reader.readLine();

// Using Scanner

Scanner in = new Scanner(System.in);

String s = in.nextLine();

int a = in.nextInt();

// Using Console

String name = System.console().readLine();

Basic Java Program

A basic program in Java will consist of at least the following components:

1. Classes & Objects
2. Methods
3. Variables

public class Demo{

public static void main(String[] args)

{ System.out.println("Hello from edureka!");}

}

Compile a Java Program

You need to save your Java Program by the name of the class containing main() method along with .java extension.

className.java

Call the compiler using javac command.

javac className

Finally, execute the program using below code:

java className

Flow Of Control

Iterative Statements

[**Iterative statements**](https://www.edureka.co/blog/java-tutorial/#control) are used when you need to repeat a set of statements until the condition for termination is not met.

// for loop

for (condition) {expression}

// for each loop

for (int i: someArray) {}

// while loop

while (condition) {expression}

// do while loop

do {expression} while(condition)

**Generating a Fibonacci series.**

for (i = 1; i <= n; ++i)

{System.out.print(t1 + " + ");

int sum = t1 + t2;

t1 = t2;

t2 = sum;}

**Creating a pyramid pattern.**

k = 2\*n - 2;

for(i=0; i<n; i++)

{ for(j=0; j<k; j++){System.out.print(" ");}

k = k - 1;

for(j=0; j<=i; j++ ){System.out.print("\* ");}

System.out.println(); }

Decisive Statements

[**Selection statements**](https://www.edureka.co/blog/java-tutorial/#control) used when you need to choose between alternative actions during execution of the program.

//if statement

if (condition) {expression}

//if-else statement

if (condition) {expression} else {expression}

//switch statement

switch (var)

{ case 1: expression; break; default: expression; break; }

**Checking the given number is prime or not.**

if (n < 2) { return false; }

for (int i=2; i <= n/i; i++)

{if (n%i == 0) return false;}

return true;

**Finding the factorial using recursion function.**

int factorial(int n)

{

if (n == 0)

{return 1;}

   else

{return(n \* factorial(n-1));}

 }

Java Arrays

Single Dimensional (1-D)

[**Single Dimensional or 1-D array**](https://www.edureka.co/blog/java-array/) is a type of linear array in which elements are stored in a continuous row.

// Initializing

type[] varName= new type[size];

// Declaring

type[] varName= new type[]{values1, value2,...};

**Creating an array with random values.**

double[] arr = new double[n];

for (int i=0; i<n; i++)

{a[i] = Math.random();}

**Searching the max value in the array.**

double max = 0;

for(int i=0; i<arr.length(); i++)

{ if(a[i] > max) max = a[i]; }

**Reversing an array.**

for(int i=0; i<(arr.length())/2; i++)

{ double temp = a[i];

a[i] = a[n-1-i];

a[n-1-i] = temp;

}

Multi Dimensional (2-D)

[**Two Dimensional or 2-D array**](https://www.edureka.co/blog/java-array/#MultidimensionalArraysinJava) is an array of an array where elements are stored in rows and columns.

// Initializing

datatype[][] varName  =  new dataType[row][col];

// Declaring

datatype[][] varName  =  {{value1, value2....},{value1, value2....}..};

**Transposing a matrix.**

for(i = 0; i < row; i++)

{ for(j = 0; j < column; j++)

{ System.out.print(array[i][j]+" "); }

System.out.println(" ");

}

**Multiplying two matrices.**

for (i = 0; i < row1; i++)

{ for (j = 0; j < col2; j++)

{ for (k = 0; k < row2; k++)

{ sum = sum + first[i][k]\*second[k][j]; }

 multiply[i][j] = sum;

sum = 0;

 }

}

Java Strings

Creating a String

[**String**](https://www.edureka.co/blog/java-string/) in Java is an object that represents a sequence of char values. A String can be created in two ways:

1. Using a literal
2. Using ‘new’ keyword

String str1 = “Welcome”; // Using literal

String str2 = new String(”Edureka”); // Using new keyword

The java.lang.String class implements Serializable, Comparable and CharSequence interfaces. Since the String object is immutable in nature Java provides two utility classes:

1. *StringBuffer:*It is a mutable class that is thread-safe and synchronized.
2. *StringBuilder:* It is a mutable class that is not thread-safe but is faster and is used in a single threaded environment.

String Methods

Few of the most important and frequently used String methods are listed below:

str1==str2 //compares address;

String newStr = str1.equals(str2); //compares the values

String newStr = str1.equalsIgnoreCase() //compares the values ignoring the case

newStr = str1.length() //calculates length

newStr = str1.charAt(i) //extract i'th character

newStr = str1.toUpperCase() //returns string in ALL CAPS

newStr = str1.toLowerCase() //returns string in ALL LOWERvCASE

newStr = str1.replace(oldVal, newVal) //search and replace

newStr = str1.trim() //trims surrounding whitespace

newStr = str1.contains("value"); //check for the values

newStr = str1.toCharArray(); // convert String to character type array

newStr = str1.IsEmpty(); //Check for empty String

newStr = str1.endsWith(); //Checks if string ends with the given