CORE JAVA CHEATSHEET

Learn JAVA from experts at www.edureka.co

Java Programming

Java is a high level, general purpose programming language that produces software for multiple platforms. It was developed by James Gosling in 1991 and released by Sun Microsystems in 1996 and is currently owned by Oracle.

Primitive Data Types

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..
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

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

Java Variables

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

Java Methods

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

Data Type Conversion

```
// 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

```
// 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();
```

Iterative Statements

```
// for loop
for (condition) {expression}

// for each loop
for (int i: someArray) {}

// while loop
while (condition) {expression}

// do while loop
do {expression} while(condition)
```

Fibonacci series

```
for (i = 1; i <= n; ++i)
 {
   System.out.print(t1 + " + ");
   int sum = t1 + t2; t1 = t2;
   t2 = sum;
 }
```

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

```
//if statement
if (condition) {expression}

//if-else statement
if (condition) {expression} else {expression}

//switch statement
switch (var) { case 1: expression; break;
default: expression; break; }
```

Prime Number

```
if (n < 2)
 {
   return false;
 }
for (int i=2; i <= n/i; i++)
 {
  if (n%i == 0) return false;
 }
return true;
```

Factorial of a Number

```
int factorial(int n)
{
   if (n == 0)
      {return 1;}
   else
      {
        return(n * factorial(n-1));
      }
}
```

Arrays In Java

1 - Dimensional

```
// Initializing
type[] varName= new type[size];

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

Array with Random Variables

```
double[] arr = new double[n];
for (int i=0; i<n; i++)
{a[i] = Math.random();}
```

Maximum value in an 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 Arrays

```
// 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 String using literal
String str1 = “Welcome”;

// Creating String using new keyword
String str2 = new String(”Edureka”);
```

String Methods

```
str1==str2 //compare the address;
String newStr = str1.equals(str2); //compares the values
String newStr = str1.equalsIgnoreCase() //
newStr = str1.length() //calculates length
newStr = str1.charAt(i) //extract i'th character
newStr = str1.toUpperCase() //returns string in ALL CAPS
```

JAVA OOP CHEAT SHEET

Learn **JAVA** from experts at https://www.edureka.co

Object Oriented Programming in Java

Java is an Object Oriented Programming language that produces software for multiple platforms. An object-based application in Java is concerned with declaring classes, creating objects from them and interacting between these objects.

Java Class

```
class Test {
        // class body
        member variables
        methods
 }
```

Java Object

```
//Declaring and Initializing an object
Test t = new Test();
```

Constructors

Default Constructor

```
class Test{
 /* Added by the Java Compiler at the Run Time
 public Test(){
 }
 */
 public static void main(String args[]) {
   Test testObj = new Test();
  }
}
```

Parameterized Constructor

```
public class Test {
   int appId;
   String appName;
//parameterized constructor with two parameters
Test(int id, String name){
   this.appId = id;
   this.appName = name;
}
void info(){
   System.out.println("Id: "+appId+" Name: "+appName);
}

public static void main(String args[]){
   Test obj1 = new Test(11001,"Facebook");
   Test obj2 = new Test(23003,"Instagram");
   obj1.info();
   obj2.info();
 }
}
```

Inheritance

Single Inheritance

```
Class A {
  //your parent class code
}
Class B extends A {
   //your child class code
}
```

Multi Level Inheritance

```
Class A {
  //your parent class code
}
Class B extends A {
   //your code
}
Class C extends B {
    //your code
}
```

Hybrid Inheritance

```
       A                               A
      / \                              |
     B   C           (OR)              B
    /     \                           / \
   D       E                         C   D
```

Hierarchical Inheritance

```
Class A {
  //your parent class code
}

Class B extends A {
   //your child class code
}

Class C extends A {
    //your child class code
}
```

Multiple Inheritance

```
Class A {
  //your parent class code
}
Class B {
   //your parent class code
}
Class C extends A,B {
    //your child class code
}
```

Polymorphism

Compile Time Polymorphism

```
class Calculator {
  static int add(int a, int b){
  return a+b;
 }
static double add( double a, double b){
  return a+b;
 }
public static void main(String args[]){
  System.out.println(Calculator.add(123,17));
  System.out.println(Calculator.add(18.3,1.9));
 }
}
```

Run Time Polymorphism

```
public class Mobile{
void sms(){System.out.println("Mobile class");}
}

//Extending the Mobile class
public class OnePlus extends Mobile{
//Overriding sms() of Mobile class
 void sms(){
  System.out.println(" OnePlus class");
 }

public static void main(String[] args) {
  OnePlus smsObj= new OnePlus();
  smsObj.sms();
 }
}
```

Abstraction

Abstract Class

```
public abstract class MyAbstractClass
 {
    public abstract void abstractMethod();
    public void display(){
     System.out.println("Concrete method");
  }
 }
```

Interface

```
//Creating an Interface
public interface Bike { public void start(); }
//Creating classes to implement Bike interface
class Honda implements Bike{
  public void start() {
   System.out.println("Honda Bike");
} }
class Apache implements Bike{
  public void start() {
    System.out.println("Apache Bike");
} }
class Rider{
  public static void main(String args[]){
   Bike b1=new Honda();
   b1.start();
   Bike b2=new Apache();
   b2.start();
} }
```

Encapsulation

```
public class Artist {
   private String name;
   //getter method
   public String getName() { return name; }
   //setter method
   public void setName(String name) { this.name = name; }
  }
public class Show{
  public static void main(String[] args){
   //creating instance of the encapsulated class
   Artist s=new Artist();
```

Modifiers in Java

Access Modifiers

Scope	Private	Default	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes

Non - Access Modifiers

Type	Scope
Static	Makes the attribute dependent on a class

JAVA STRING CHEAT SHEET

Learn JAVA from experts at http://www.edureka.co

Java Strings

In **Java**, a **string** is an object that represents a sequence of characters. The **java.lang.String** class is used to create **string** object. **String** contains an immutable sequence of Unicode characters.

Creating a String

```
String str1 = “Welcome”;
// Using literal String
str2 = new String(”Edureka”);
 // Using new keyword
```

Immutable Strings

```
class Stringimmutable
{
public static void main(String args[])
{
 String s="JavaStrings";
 s.concat(" CheatSheet");
 System.out.println(s);
 }
}
```

Methods of Strings

```
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
suffix
```

Programs

Removing Trailing spaces from string

```
int len = str.length();
for( ; len > 0; len--) {
 if( ! Character.isWhitespace(
str.charAt( len - 1)))
 break;
}
return str.substring( 0, len);
```

Finding Duplicate characters in a String

```
public void countDupChars{
Map<Character, Integer> map = new HashMap
<Character, Integer>();
//Convert the String to char array
char[] chars = str.toCharArray();
Set<Character> keys = map.keySet();
//Obtaining set of keys
public static void main(){
System.out.println("String: Edureka");
obj.countDupChars("Edureka");
System.out.println("\nString:
StringCheatSheet");
obj.countDupChars("StringCheatSheet");
}
}
```

String Joiner Class

```
StringJoiner mystring = new
StringJoiner("-");
// Passing Hyphen(-) as delimiter
mystring.add("edureka");
// Joining multiple strings by using
add() method
mystring.add("YouTube");
```

String reverse using Recursion

```
String str = "Welcome to Edureka";
String reversed=reverseString(str);
ReturnreverseString(str.substring(1))
+ str.charAt(0);
//Calling Function Recursively
```

String reverse

```
String str;
System.out.println("Enter your
username: ");
String reversed = reverseString(str);
// Reversing a String
return reverseString(str.substring(1))
+ str.charAt(0);
//Calling Function Recursively
```

String Pool

```
String str1 = "abc";
String str2 = "abc";
System.out.println(str1 == str2);
System.out.println(str1 == "abc");
```

String Conversions

String to Int Conversion

```
String str="123";
int inum1 = 100;
int inum2 =
Integer.parseInt(str);
// Converting a string to int
```

Int to String Conversion

```
int var = 111;
String str =
String.valueOf(var);
System.out.println(555+str);
// Conversion of Int to String
```

String to Double Conversion

```
String str = "100.222";
double dnum = Double.parseDouble(str);
//displaying the value of variable dnum
```

Double to String Conversion

```
double dnum = 88.9999; //double value
String str = String.valueOf(dnum);
//conversion using valueOf() method
```

String vs String Buffer

It is immutable	It is mutable
String class overrides the equals() method of Object class..	StringBuffer class doesn't override the equals() method of Object class.

String Buffer vs Builder

StringBuffer is *synchronized* i.e. thread safe.	StringBuilder is *non-synchronized* i.e. not thread safe.
StringBuffer is *less efficient* than StringBuilder as it is Synchronized.	StringBuilder is *more efficient* than StringBuffer as it is not synchronized.