Java Programming Language Cheatsheet

Updated on 27 January 2024 by Husaif Sayyed

Java is a versatile and widely-used programming language that has been a cornerstone of software development for decades. Whether you're a seasoned Java developer or a beginner getting started with the language, having a cheatsheet at your disposal can be incredibly handy. This Java cheat sheet aims to provide a quick reference guide to essential Java concepts, syntax, and best practices.

Links

Java Official Website

Download PDF

★ Want More Cheat Sheet

Hello World! Java Program

Create a file called hello.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Use javac hello.java to compile the code, and then run java Hello

Basics

Comments

```
// Single-line comment

/*
  Multi-line
  comment
*/

/**
 * Javadoc comment
 */
```

Variables

```
int age = 25;

double price = 19.99;

char grade = 'A';

String name = "John";

boolean isJavaFun = true;
```

Data Types

Primitive Types: int, double, char, boolean

Reference Types: String, Arrays, Classes

Operators

```
int sum = a + b;
int difference = a - b;
int product = a * b;
int quotient = a / b;
int remainder = a % b;
```

Control Flow

Conditional Statements

```
if (condition) {
    // code to execute if condition is true
} else if (anotherCondition) {
    // code to execute if anotherCondition is true
} else {
    // code to execute if none of the conditions are t
}
```

Switch Statement

```
switch (variable) {
    case value1:
        // code to execute if variable equals value1
        break;
    case value2:
        // code to execute if variable equals value2
        break;
    default:
        // code to execute if variable doesn't match a
}
```

Loops

for Loop

```
for (int i = 0; i < 5; i++) {
    // code to repeat five times
}
```

while Loop

```
while (condition) {
    // code to repeat as long as condition is true
}
```

Do While Loop

```
do {
    // code to execute at least once, then repeat while
} while (condition);
```

Break and Continue Statements

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break; // exit the loop when i equals 5
    }
    if (i == 3) {
        continue; // skip the rest of the loop and cont
    }
    // code inside the loop
}
```

Functions and Methods

Method Declaration

```
public returnType methodName(parameterType parameterNa
    // method body
}
```

Return Statement

```
public int add(int a, int b) {
    return a + b;
}
```

Parameters

```
public void greet(String name) {
    System.out.println("Hello, " + name + "!");
}
```

Overloading

```
public int add(int a, int b) {
    return a + b;
}

public double add(double a, double b) {
    return a + b;
}
```

Arrays

Declaration and Initialization

```
int[] numbers = new int[5];
int[] nums = {1, 2, 3, 4, 5};
```

Accessing Elements

```
int firstElement = nums[0];
```

Multi-dimensional Arrays

```
int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

Object-Oriented Programming (OOP)

Classes and Objects

```
public class Car {
    String brand;
    int year;

    public Car(String brand, int year) {
        this.brand = brand;
        this.year = year;
    }
}
```

Constructors

```
public class Person {
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

Inheritance

```
public class Animal {
    void eat() {
        System.out.println("Animal is eating");
    }
}

public class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking");
    }
}
```

Polymorphism

```
public class Shape {
    void draw() {
        System.out.println("Drawing a shape");
    }
}

public class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle");
    }
}
```

Encapsulation

```
public class Student {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Abstraction

```
abstract class Shape {
    abstract void draw();
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle");
    }
}
```

Exception Handling

try-catch Blocks

```
try {
    // code that may throw an exception
} catch (ExceptionType e) {
    // handle the exception
} finally {
    // code to execute regardless of whether an except
}
```

Custom Exceptions

```
class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}
```

Collections Framework

Lists, Sets, Maps

```
List<String> list = new ArrayList<>();
Set<Integer> set = new HashSet<>();
Map<String, Integer> map = new HashMap<>();
```

Iterators

```
Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
    String element = iterator.next();
    // code to process each element
}
```

ArrayList, LinkedList

```
List<String> arrayList = new ArrayList<>();
List<String> linkedList = new LinkedList<>();
```

HashSet, TreeSet

```
Set<String> hashSet = new HashSet<>();
Set<String> treeSet = new TreeSet<>();
```

HashMap, TreeMap

```
Map<String, Integer> hashMap = new HashMap<>();
Map<String, Integer> treeMap = new TreeMap<>();
```

File I/O

Reading from File

```
try (BufferedReader br = new BufferedReader(new FileRe
    String line;
    while ((line = br.readLine()) != null) {
        // process each line
    }
} catch (IOException e) {
    // handle exception
}
```

Writing to File

```
try (BufferedWriter bw = new BufferedWriter(new FileWr
    bw.write("Hello, World!");
} catch (IOException e) {
    // handle exception
}
```

Concurrency

Threads

```
class MyThread extends Thread {
    public void run() {
        // code to be executed in the new thread
    }
}

MyThread thread = new MyThread();
thread.start();
```

Synchronization

```
class Counter {
    private int count = 0;

    public synchronized void increment() {
        count++;
    }
}
```

Thread Pools

```
ExecutorService executor = Executors.newFixedThreadPool(5);
executor.submit(() -> {
    // code to be executed in a thread from the pool
});
executor.shutdown();
```

Lambda Expressions

Syntax

```
interface MyFunctionalInterface {
    void myMethod();
}

MyFunctionalInterface myFunction = () -> {
    // implementation of myMethod
};
```

Functional Interfaces

```
@FunctionalInterface
interface Calculator {
    int calculate(int a, int b);
}

Calculator addition = (a, b) -> a + b;
```

Streams

```
List<String> strings = Arrays.asList("one", "two", "three");

strings.stream()
    .filter(s -> s.length() > 3)
    .map(String::toUpperCase)
    .forEach(System.out::println);
```

Congratulations on reaching the end of this Java Programming Language Cheatsheet! This resource is designed to make your coding experience easy and efficient. Feel free to bookmark this page or download the PDF for future reference. Happy Python Programming!