Java Cheat Sheet

1. Basic Syntax

Hello World Program:

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

Comments:

- **Single-line comment**: `// This is a comment`
- **Multi-line comment**: `/* This is a comment */`
- **Doc comment**: `/** This is a doc comment */`

---

2. Variables and Data Types

Primitive Data Types:

```
java
Copy code
int x = 10;          // Integer
double y = 5.5;      // Floating-point number
char c = 'A';        // Character
boolean isTrue = true; // Boolean
long bigNumber = 1234567890L; // Long integer
float f = 3.14f;     // Float
byte b = 100;        // Byte
short s = 200;       // Short
```

String:

```
java
Copy code
String str = "Hello, Java!";
```

---

3. Control Flow

If-Else:

```
java
Copy code
if (x > 10) {
    System.out.println("x is greater than 10");
} else {
    System.out.println("x is not greater than 10");
}
```

Switch:

```
java
Copy code
int day = 3;
```

```
switch(day) {
    case 1: System.out.println("Monday"); break;
    case 2: System.out.println("Tuesday"); break;
    case 3: System.out.println("Wednesday"); break;
    default: System.out.println("Invalid day");
}
```

For Loop:

```
java
Copy code
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

While Loop:

```
java
Copy code
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

Do-While Loop:

```
java
Copy code
int i = 0;
do {
    System.out.println(i);
    i++;
} while (i < 5);
```

---

4. Methods (Functions)

Defining a Method:

```
java
Copy code
public static int add(int a, int b) {
    return a + b;
}
```

Calling a Method:

```
java
Copy code
int sum = add(5, 10);
```

Method Overloading:

```
java
Copy code
public static void print(int num) {
    System.out.println(num);
}

public static void print(String str) {
    System.out.println(str);
}
```

Return Type:

```
java
Copy code
public static String greet(String name) {
    return "Hello, " + name;
}
```

---

5. Arrays

Array Declaration and Initialization:

```
java
Copy code
int[] arr = new int[5];      // Declare array of integers
arr[0] = 10;                  // Assign value
```

Array Initialization (Short Syntax):

```
java
Copy code
int[] arr = {1, 2, 3, 4, 5};
```

Iterating over an Array:

```
java
Copy code
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
```

Enhanced for loop (foreach):

```
java
Copy code
for (int num : arr) {
    System.out.println(num);
}
```

---

6. Classes and Objects

Class Definition:

```
java
Copy code
public class Car {
    String model;
    int year;

    // Constructor
    public Car(String model, int year) {
        this.model = model;
        this.year = year;
    }

    // Method
    public void displayInfo() {
        System.out.println(model + " " + year);
    }
}
```

Creating an Object:

```
java
Copy code
Car myCar = new Car("Toyota", 2020);
myCar.displayInfo();
```

Constructor Overloading:

```
java
Copy code
public class Car {
    String model;
    int year;

    // Constructor
    public Car(String model) {
        this.model = model;
    }

    // Overloaded Constructor
    public Car(String model, int year) {
        this.model = model;
        this.year = year;
    }
}
```

---

7. Inheritance

Basic Inheritance:

```
java
Copy code
class Animal {
    void makeSound() {
        System.out.println("Animal makes sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Dog barks");
    }
}

Dog dog = new Dog();
dog.makeSound();  // Outputs: Dog barks
```

---

8. Polymorphism

Method Overriding:

```
java
Copy code
class Animal {
    void makeSound() {
        System.out.println("Animal makes sound");
    }
}
```

```
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog barks");
    }
}

Animal a = new Dog();
a.makeSound();  // Outputs: Dog barks
```

Method Overloading:

```
java
Copy code
class Printer {
    void print(int num) {
        System.out.println("Printing number: " + num);
    }

    void print(String text) {
        System.out.println("Printing text: " + text);
    }
}
```

---

9. Interfaces and Abstract Classes

Interface:

```
java
Copy code
interface Animal {
    void sound();
}

class Dog implements Animal {
    @Override
    public void sound() {
        System.out.println("Bark");
    }
}
```

Abstract Class:

```
java
Copy code
abstract class Animal {
    abstract void sound();  // Abstract method

    void eat() {
        System.out.println("Eating...");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Bark");
    }
}
```

---

10. Exception Handling

Try-Catch:

```
java
Copy code
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Error: " + e.getMessage());
}
```

Finally Block:

```
java
Copy code
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Error: " + e.getMessage());
} finally {
    System.out.println("This will always execute.");
}
```

Throwing Exceptions:

```
java
Copy code
public static void validateAge(int age) {
    if (age < 18) {
        throw new IllegalArgumentException("Age must be 18 or older.");
    }
}
```

---

11. Collections Framework

List:

```
java
Copy code
import java.util.*;

List<String> names = new ArrayList<>();
names.add("Alice");
names.add("Bob");
```

Set:

```
java
Copy code
Set<Integer> numbers = new HashSet<>();
numbers.add(1);
numbers.add(2);
```

Map:

```
java
Copy code
Map<String, Integer> map = new HashMap<>();
map.put("One", 1);
map.put("Two", 2);
```

Queue:

```
java
Copy code
Queue<Integer> queue = new LinkedList<>();
queue.add(1);
queue.add(2);
```

---

12. Lambda Expressions

Basic Syntax:

```
java
Copy code
// Functional Interface
@FunctionalInterface
interface MyFunction {
    void printMessage(String msg);
}

public class Main {
    public static void main(String[] args) {
        MyFunction f = (msg) -> System.out.println(msg);
        f.printMessage("Hello from Lambda!");
    }
}
```

---

13. Streams API

Stream Operations:

```
java
Copy code
import java.util.*;
import java.util.stream.*;

List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
int sum = numbers.stream().mapToInt(Integer::intValue).sum();
System.out.println("Sum: " + sum);
```

---

14. Multithreading

Creating a Thread:

```
java
Copy code
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running");
    }
}

public class Main {
    public static void main(String[] args) {
        MyThread t = new MyThread();
        t.start();
    }
}
```

Runnable Interface:

```
java
Copy code
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Thread is running");
    }
}

public class Main {
    public static void main(String[] args) {
        MyRunnable r = new MyRunnable();
        Thread t = new Thread(r);
        t.start();
    }
}
```