JAVA CHEATSHEET

Basic structure of Java Program

```
public class Main {


    public static void main(String[] args) {

        System.out.println("Hello, World!");

    }

}
```

Comments

Comments are used In Java to make the code more readable and understandable for developers.

Single Line Comments

```
// This is a single line comment
```

Multi-line comments

```
/* This is a
multi line
comment */
```

Data Types

Data types are used in Java to classify the various types of data that can be stored in a variable. There are two types of datatypes in Java:

Primitive Data Types

- **bool:** Boolean data type represents one bit of information either true or false.
- **char:** The char data type is a single 16-bit Unicode character.
- **byte:** Byte data type is an 8-bit signed two's complement integer.

- **short:** Short data type is a 16-bit signed two's complement integer.
- **int:** It is a 32-bit signed two's complement integer.
- **long:** Long data type is a 64-bit signed two's complement integer.
- **float:** The float data type is a single-precision 32-bit IEEE 754 floating point.
- **double:**The double data type is a double-precision 64-bit IEEE 754 floating-point.

Data Type	Size	Range
bool	1 bit	true, false
char	2 byte	0 to 255
byte	1 byte	-128 to 127
short	2 byte	-32,768 to 32,767
int	4 byte	-2,147,483,648 to 2,147,483,647
long	8 byte	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 byte	upto 7 decimal digits

If Statement

The if statement is used to execute a block of code if a given condition is true.

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

If...else statement

The If...else statement is used to execute a block of code if a specified condition is true and another block of code if the condition is false.

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

if…elif…else Statement

Java if-elif-else statement executes a block of code among multiple possibilities.

```
if (condition) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2
is true
} else {
  // block of code to be executed if the condition1 is false and condition2
is false
}
```

Switch

The switch statement is used to select one of many code blocks to be executed.

```
switch (expression) {
  case value1:
    // block of code
    break;
  case value2:
    // block of code
    break;
  default:
    // block of code
}
```

For Loop

A for loop is used to execute a piece of code a specified number of times.

```
for (initialization; testExpression; increment/decrement) {
   // block of code
}
```

While Loop

The while loop is used to execute a block of code as long as a specified condition is true.

```
while (condition) {
  // block of code
}
```

Example:

```
public class Main {
  public static void main(String[] args) {
    int i = 1;
    while (i <= 8) {
      System.out.println(i);
      i++;
    }
  }
}
```

Do...While Loop

The do-while loop is similar to the while loop. This loop would execute its statements at least once, even if the condition fails for the first time.

```
do {
  // block of code
} while (condition);
```

Example:

```
public class Main {
  public static void main(String[] args) {
    int i = 0;
    do {
      System.out.println(i);
      i++;
    } while (i < 5);
  }
}
```

Break Statement

The break statement is used to break out of the loop in which it is encountered. The break statement is used inside loops or switch statements in C programming.

Example:

```
public class Main {
  public static void main(String[] args) {
    int i;
    for (i = 0; i < 10; i++) {
      if (i == 6) {
        break;
      }
      System.out.println(i);
    }
  }
}
```

Continue Statement

The continue statement skips the loop's current iteration and proceeds to the next one.

Example:

```
public class Main {
  public static void main(String[] args) {
    int i;
    for (i = 1; i < 10; i++) {
      if (i == 3) {
       continue;
      }
      System.out.println(i);
    }
  }
}
```

Arrays

Arrays in Java are used to store multiple values in a single variable.

```
String[] companies = {"Goggle", "Facebook", "Microsoft"};
```

Accessing Elements of an Array

```
public class Main {
  public static void main(String[] args) {
    String[] companies = {"Goggle", "Facebook", "Microsoft"};
    System.out.println(companies[1]);
  }
}
```

OOP

Java is an object-oriented programming language. The core concept of the object-oriented programming is to break complex problems into smaller objects.

Class

A class is a blueprint for creating objects.

Create a Class

To create a class, use the keyword `class`:

```
public class Main {

  int x = 5;

}
```

Objects

An object is called an instance of a class.

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Methods

A method is a block of code that performs a specific task.

Declaring a Method

The syntax to declare a method is:

```
returnType methodName() {
  // method body
}
```

Calling a Method

```
addNumbers();
```

Example:

```
class Main {

  // create a method

  public int addNumbers(int a, int b) {

    int sum = a + b;

    // return value

    return sum;

  }

  public static void main(String[] args) {

    int num1 = 25;

    int num2 = 15;

    // create an object of Main

    Main obj = new Main();

    // calling method

    int result = obj.addNumbers(num1, num2);

    System.out.println("Sum is: " + result);

  }

}
```

Strings

Java strings are a sequence of characters that are enclosed by double quotes.

```
public class Main {
  public static void main(String[] args) {
    String name = "Messi";
    System.out.println(name);
  }
}
```

String Concatenation

```
public class Main {
  public static void main(String[] args) {
    String firstName = "Lionel";
    String lastName = "Messi";
    System.out.println(firstName + " " + lastName);
  }
}
```

String Methods

length()

The length method returns the length of a string.

Example:

```
public class Main {
  public static void main(String[] args) {
    String greeting = "Hello Java";
    System.out.println(greeting.length());
  }
}
```

toUpperCase() and toLowerCase()

The toUpperCase() and toLowerCase() methods are used to convert a string to uppercase or lowercase letters.

Example:

```
public class Main {
  public static void main(String[] args) {
    String greeting = "Hello Java";
    System.out.println(greeting.toUpperCase());
    System.out.println(greeting.toLowerCase());
  }
}
```

Inheritance

Inheritance is one of the key features of OOP that allows us to create a new class from an existing class.

The `extends` keyword is used to perform inheritance in Java.

Example:

```
class Vehicle {
  protected String brand = "Yamaha";
  public void honk() {
    System.out.println("Bhrum, bhruum!");
  }
}
class Bike extends Vehicle {
  private String modelName = "R15";
  public static void main(String[] args) {
    Bike myFastBike = new Bike();
    myFastBike.honk();
    System.out.println(myFastBike.brand + " " + myFastBike.modelName);
  }
}
```

Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Example:

```
class Bird {

  public void birdSound() {

    System.out.println("The bird makes a sound");

  }

}

class Owl extends Bird {

  public void birdSound() {

    System.out.println("The owl sound is: hoot");

  }

}

class Peacock extends Bird {

  public void birdSound() {

    System.out.println("The peacock sound is: scream");

  }

}

class Main {

  public static void main(String[] args) {

    Bird myBird = new Bird();

    Bird myOwl = new Owl();

    Bird myPeacock = new Peacock();

    myBird.birdSound();

    myOwl.birdSound();

    myPeacock.birdSound();

  }

}
```