JAVA CHEATSHEET

Contents :

Introduction to JAVA Programming

Operators :

Operators are basically symbols that help us perform all kinds of operations in our program. Beginning with addition, subtraction to comparison and also bitwise operators.

1. Arithmetic operators are addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^), modulo (%)
2. Relational operators are <, >, ==, <=, >=
3. Logical operators - AND (&), OR (|), NOT (!)
4. Assignment operator is equal to sign (=)

Reading Inputs :

We can take inputs from the console and use them in our program. This can be done using the Scanner class in the java's utility package.

```
import java.util.Scanner;
public class hello {
   public static void main(String[] args) {
       // Creating an instance, called "input" of the Scanner class.
       Scanner input = new Scanner(System.in);
       /** Scanner class provides us with a wide variety of function to take
        all kinds of input from the user. **/
       int a = input.nextInt();
       float b = input.nextFloat();
       double c = input.nextDouble();
       // the next and nextLine is used to input Strings
       // next is used to input the characters before the first space is encountered
       // nextLine is used to enter all characters before line is changed
       String d = input.next();
       String e = input.nextLine();
   }
}
```

Loops :

The basic idea about loops is to execute a set of statements, several times.

Each loop has 3 parts.

- Initialization - The start of the loop
- Condition checking - The condition is checked at every iteration, and the loop stops when the condition becomes false.
- Updation - The counter of the loop is updated at every iteration, according to the updation rule.

- Program demonstrating the use of for loop with conditional statements

```
import java.util.Scanner;
public class hello {

   public static void main(String[] args) {
       // Let's look at some loops
       // Write a code to print the fibonacci series
       Scanner input = new Scanner(System.in);
       System.out.println("How many terms of the fibonacci series do you want ?");
       int n = input.nextInt();
       if (n == 0)
           System.out.println("Nothing is printed");
       else {
           System.out.print("0 ");
           int fib1 = 0;
           int fib2 = 1;
           int fib = 1;
           for(int i = 1; i < n; i++){
               System.out.print(fib + " ");
               fib = fib1 + fib2;
               fib1 = fib2;
               fib2 = fib;
           }
       }
   }
```

```
}
```

- ## While Loop and Nested if-else

```
import java.util.Scanner;
public class hello {
   public static void main(String[] args) {
       // Time for some while loop and nested if-else
       /** We will write a program to print if a number is odd or even,
       and terminate if the user enters -1. **/
       int j = 0;
       Scanner input = new Scanner(System.in);
       while(j != -1) {
           System.out.println("Please enter a number ");
           j = input.nextInt();
           if (j != -1) {
               if (j % 2 == 0)
                   System.out.println("It's a even number");
               else
                   System.out.println("It's a odd number");
           }
           else
               System.out.println("The end");
       }
   }
}
```

Conditional Statements :

These coding constructs are used to run a set of lines if, and only if, a specific condition is met.

- ## If-else constructs

```
public class hello {
   public static void main(String[] args) {
       // Let's try out some conditional Statements
       int marks = 60;
       if (marks >= 90)
```

```
            System.out.println("Amazing, distinction grade");
        else if (marks >= 75 & marks < 90)
            System.out.println("Awesome, 1st grade");
        else if (marks >= 60 & marks < 75)
            System.out.println("Good, 2nd grade");
        else
            System.out.println("You just passed, need to work hard");
    }
}
```

- ## Switch Case and Nested Switch Case :

```
import java.util.Scanner;
public class Examples {
    public static void main(String[] args) {
        // Time to play with some switch and nested switch cases...
        /** In this question we will decide if a number is good or not. 1 and 2
         are good numbers, but if it's 3, then we ask them to enter a different number **/
        Scanner input = new Scanner(System.in);
        System.out.print("Please enter a number between 1-3 ");
        int a = input.nextInt();
        switch(a) {
            case 1:
                System.out.println("1 is a beautiful number");
                break;
            case 2:
                System.out.println("2 is an awesome number. It's also the only even prime");
                break;
            case 3:
                System.out.println("Oops. Please re-enter a number between 4 and 5");
                int b = input.nextInt();
                switch(b) {
                    case 4:
                        System.out.println("Superb choice, you got some some taste");
                        break;
                    case 5:
                        System.out.println("Great. 5 is also my favourite");
                        break;
                    default:
                        System.out.println("Wrong choice");
```

```
            }
        default:
            System.out.println("Wrong choice");
    }
  }
}
```

Type Casting :

It is basically a way to convert a variable in one data type to another.

- Implicit Type Cast - Happens automatically when you store a less precise (or smaller) datatype to a more precise (or larger) datatype.
- Explicit Type Cast - It needs to be done, if you want to change a larger datatype to a smaller one.

```
public class hello {
  public static void main(String[] args) {
      // Let's look at some type casting
      // Implicit type Casting
      int a = 5;
      float b = a;
      // Explicit Type Casting
      double c = 3.14159265;
      float d = (float) c;
  }
}
```

Arrays :

Arrays is a collection of variables of similar data types. The memory locations of the block are continuous. Suppose an array of int, of size 5. The size of the array is 4 bytes x 5 = 20 bytes. Static array has a fixed size, whereas the size of a dynamic array can be changed as and when required.

- *Program to show basics of Arrays.*

```
public class hello {
   public static void main(String[] args) {
       // Let's learn some arrays...
       // Let's declare an array of doubles.
       double[] a = {2.3, 4.5, 4.56, 3.43, 9.01};
       // Let's declare instantiate and initialize an array, all in one
       int b[] = {1, 2, 3, 6, 5};
       // Time for some array addition
       int c[] = {12, 43, 21, 34, 20};
       // Length function can be used to get the length of the array.
       for(int i = 0; i < c.length; i++)
           System.out.print(b[i] + c[i] + " ");
   }
}
```

Arrays can be 1D, 2D or multi-D. Ever heard of a matrix ? A 2D array is a matrix, where each element has 2 indices, its row index, and column index. It can be generalized to a 3D array, where every element has 3 associated indices.

- *Program to demonstrate 2D array, nested loop and jump statement*

```
public class hello {
   public static void main(String[] args) {
       // Let's create a 2D array and search for some elements
       /** First we will declare a 2D array of our own, and then using a nested loop
        we will search for the target element **/
       int [][] arr = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
       /** Target is the value that need to be found, and flag variable is just
        an indicator. If flag is true, then the value i found **/
       int target = 7;
       boolean flag = false;
       for(int i = 0; i < 3; i++) {
           for(int j = 0; j < 3; j++) {
               if (arr[i][j] == target) {
                   flag = true;
                   System.out.println("Yahoo...element found");
```

```
                /** We are using the jump statement, called break taht helps
                to break out of any loop construct or iterative process. **/
                break;
            }
        }
    }
    // Here the logical operator "NOT" is used in a very elegant manner.
    if(!flag)
        System.out.println("Oops, not found");
  }
}
```

Strings :

Similar to arrays, String is a collection of characters. It's an object in JAVA. We have several String manipulation functions that help us work with Strings easily.

- Program for String declaration and some basic, but important comparison functions

```
public class hello {
  public static void main(String[] args) {
    // Time for some Strings
    // Declaring a String
    String name = "Harry Potter";
    // Creating an instance of the String using the new keyword
    String company = new String();
    company = "Harry Potter";
    // Let's do some comparison using different methods
    if (name == company)
        System.out.println("They are the one and the same");
    else
        System.out.println("They are not the same");
    // Equals function returns a boolean value.
    boolean a = name.equals(company);
    /** The compareTo function returns the difference in the ASCII values
    In this case it is, H - h is -32. SO value of j is -32 **/
```

```
        int j = name.compareTo("harry potter");
        System.out.println(a + " " + j);
    }
}
```

- ## Demonstration of String Manipulation functions using Array of Strings

```
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        // Let's perform some String manipulations on an array of Strings...
        Scanner input = new Scanner(System.in);
        String arr[] = new String[4];
        for(int i = 0; i < 4; i++) {
            System.out.println("Please input the Name at index " + i);
            arr[i] = input.nextLine();
        }
        // Print the substring of every name starting from index 2, till the end in upper case.
        for(int i = 0; i < 4; i++)
            /** This is a very useful way to perform the operations. We can use 2,
            maybe more functions in a single line using the dot operator. **/
            System.out.println(arr[i].substring(2).toUpperCase());
    }
}
```