Java Cheatsheet

Boilerplate

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

Showing Output

It will print something to the output console.

```
System.out.println([text])
```

Taking Input

It will take string input from the user

```
import  java.util.Scanner; //import scanner class
// create an object of Scanner class
Scanner input = new Scanner(System.in);
// take input from the user
String varName = input.nextLine();
```

Comments

A comment is the code that is not executed by the compiler, and the programmer uses it to keep track of the code.

Single line comment

```
// It's a single line comment
```

Multi-line comment

```
/* It's a
multi-line
comment
*/
```

Variables

The eight primitives defined in Java are int, byte, short, long, float, double, boolean, and char those aren't considered objects and represent raw values.

byte

byte is a primitive data type it only takes up 8 bits of memory.

```
age = 18;
```

long

long is another primitive data type related to integers. long takes up 64 bits of memory.

```
viewsCount = 3_123_456L;
```

float

We represent basic fractional numbers in Java using the float type. This is a single-precision decimal number. Which means if we get past six decimal points, this number becomes less precise and more of an estimate.

```
price = 1.1;
```

char

Char is a 16-bit integer representing a Unicode-encoded character.

```
letter = 'A';
```

boolean

The simplest primitive data type is boolean. It can contain only two values: true or false. It stores its value in a single bit.

```
isEligible = true;
```

int

int holds a wide range of non-fractional number values.

```
var1 = 256;
```

short

If we want to save memory and byte is too small, we can use short.

short var2 = 786;

Constants

Constants are like a variable, except that their value never changes during program execution.

final float PI= 22/7;

Strings

Creating String Variable

String a = "Zain";

Method	**Use**
a.charAt(index);	To find the index of character
a.contains("Z");	To Search in the string
a.toLowerCase() , a.toUpperCase()	To convert upper or lower case
a.startswith("Z"); a.endsWith("n");	To verify wheather string start/end with particular letter.
String b=a.split("i");	To split String into sub String
replaceAll(who, with what to replace) a.replaceAll("Z","W");	To replace any character or space with the other one.
a.match()	To check weather the String matching or not.
a.indexOf("a")	Use to find the index of particular character

Object Data Type

It contain all type of data in it.

Object a=10; or Object a="Zain";

Arrays

Arrays are used to store multiple values in a single variable

Declaring an array

Declaration of an array

```
String[] var_name;  or
String var_name[]=new String[size];
```

Method	Use
arr[index]	Accessing value at particular index.
Arr.length()	Getting length of array
arr[index]="new Value";	Changing value at particular index.
String arr[] = {"2","3"}; arr[index].contains("2")	Check whether array contain value at particular index or not. In String array it work properly.

Multi-dimensional Arrays

Arrays can be 1-D, 2-D or multi-dimensional.

```
// Creating a 2x3 array (two rows, three columns)
int[2][3] matrix = new int[2][3];
matrix[0][0] = 10;
// Shortcut
int[2][3] matrix = {
{ 1, 2, 3 },
{ 4, 5, 6 }
};
```

Type Casting

Type Casting is a process of converting one data type into another

Widening Type Casting (int to double)

It means converting a lower data type into a higher

```
// int x = 45;

double var_name = x;
```

Narrowing Type Casting (double to int)

It means converting a higher data type into a lower

```
double x = 165.48

int var_name = (int)x;
```

String to integer conversion:

String a =“20”;

int b=new Integer(a); **or**

String a="12";

int b=Integer.parseInt(a);

Integer to String:

Int a=10;

String b=integer.toString(a);

Character to integer

Char c=’2’;

int s=Integer.parseInt(String.valueOf(c));

ArrayList

Declaration: ArrayList list = new ArrayList();

Methods:

List.add(1);

list.add(0, "umer");

List.contains(1);

List.remove(1);

List.clear();

List2=List.clone();

Linked List

```
LinkedList obj=new LinkedList();
```

HashSet

```
HashSet<Integer> obj=new HashSet<Integer>();

Obj.add(2);
```

(all functions similar like above but the advantage is when we will search

it will take less time then others.)

Math Class

Math class allows you to perform mathematical operations.

Methods max() method

It is used to find the greater number among the two

```
Math.max(25, 45);
```

min() method

It is used to find the smaller number among the two

```
Math.min(8, 7);
```

sqrt() method

It returns the square root of the supplied value

```
Math.sqrt(144);
```

random() method

It is used to generate random numbers

```
Math.random(); //It will produce random number b/w 0.0 and 1.0
int random_num = (int)(Math.random() * 101); //Random num b/w 0 and 100
```

Time Methods

LocalDate	LocalTime	LocalDateTime	**DateTimeFormatter**
.now()	.now()	.now()	**DateTimeFormatter obj=DateTimeFormatter.ofPattern("a");**

Date time formatter patterns

Symbol	**Meaning**	**Presentation**	**Examples**
y	year-of-era	Year	2004; 04
D	day-of-year	number	189
M/L	month-of-year	number/text	7; 07; Jul; July; J
d	day-of-month	number	10
g	modified-julian-day	number	2451334
Q/q	quarter-of-year	number/text	3; 03; Q3; 3rd quarter
W	week-of-month	number	4
E	day-of-week	text	Tue; Tuesday; T
e/c	localized day-of-week	number/text	2; 02; Tue; Tuesday; T
F	day-of-week-in-month	number	3
a	am-pm-of-day	text	PM
h	clock-hour-of-am-pm (1-12)	number	12
m	minute-of-hour	number	30

s	second-of-minute	number	55

Date Class Methods

```
Date obj=new Date();
          System.out.println("date "+obj.getDate());
          System.out.println("Month "+obj.getMonth());
          System.out.println("Day "+obj.getDay());
          System.out.println("Year "+obj.getYear());
          System.out.println("Hours "+obj.getHours());
          System.out.println("Minutes "+obj.getMinutes());
          System.out.println("Seonds "+obj.getSeconds());
```

Reguler Expressions for validation

Pattern.matches(pattern , with which we have to match);

Pattern.macthes(“[a-z]{1,4}”,”Hello”);

Different patterns.

[a-z]{1,}   [a-z A-Z]{1,}   [\d] =digits   [^abc]= any alphabet except abc letters

[M] [a-z] {5} (here first letter will must be capital M).

Email: [a-z]{2}@[gmail]+.[com]+

Quantifers:

Quantifier	Occurrence
[ ]?	0 or 1 time
[ ]*	0 to many times
[ ]+	1 to many times
{minimum , maximum}	Minimum time must maximum limit provided.

Functions / Methods

Methods are used to divide an extensive program into smaller pieces. It can be called multiple times to provide reusability to the program.

Declaration

Declaration of a method

```
returnType methodName(parameters) {
//statements
}
```

Calling a method

Calling a method

```
methodName(arguments);
```

Method Overloading

Method overloading means having multiple methods with the same name, but different parameters.

```
class Calculate
{
void sum (int x, int y)
{
System.out.println("Sum is: "+(a+b)) ;
}
void sum (float x, float y)
{
System.out.println("Sum is: "+(a+b));
}
Public static void main (String[] args)
{
Calculate calc = new Calculate();
calc.sum (5,4); //sum(int x, int y) is method is called.
calc.sum (1.2f, 5.6f); //sum(float x, float y) is called.
} }
```

Recursion

Recursion is when a function calls a copy of itself to work on a minor problem. And the function that calls itself is known as the Recursive function.

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

Condition statements

if else-if Statement

```
if (condition1) {
// Codes
}
else if(condition2) {
// Codes
}
else {
// Codes
}
```

Ternary Operator

It is shorthand of an if-else statement.

```
variable = (condition) ? True statement : False statement;


               String a=(10<5)?"It is less ":"it is Greater";
                       System.out.println(a);
```

Switch Statements

It allows a variable to be tested for equality against a list of values (cases).

```
switch(expression) {
case a:
// code block
break;
case b:
// code block
break;
default:
// code block
}
```

Iterative Statements / LOOPS

Iterative statements facilitate programmers to execute any block of code lines repeatedly and can be controlled as per conditions added by the coder.

while Loop

It iterates the block of code as long as a specified condition is True

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

for Loop

```
for (initialization; termination; increment) {
statement(s)
}
```

for-each Loop

```
for(dataType item : array) {
 ...
}
```
e.g
```
for(String a: array) {
 System.out.print(a)   }
```

do-while Loop

```
do {
```

```
// body of loop

} while(textExpression)
```

Break statement

break keyword inside the loop is used to terminate the loop

```
break;
```

Continue statement

continue keyword skips the rest of the current iteration of the loop and returns to the starting point of the loop continue;

Object-Oriented Programming

It is a programming approach that primarily focuses on using objects and classes. The objects can be any real-world entities.

class

A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

```
class ClassName {

// Fields

// Methods

// Constructors

// Blocks

}
```

object

An object is an instance of a Class.It make the class specific.like animal class and we make obj of cat, it will make it specific now.

```
className object = new className();
```

Encapsulation (Making abstract)

Encapsulation is a mechanism of wrapping the data and code acting on the data together as a single unit. In encapsulation, the variables of a class will be hidden from other classes making it **private** and can be accessed only through the methods of their current class.

```
public class Person {
private String name; // using private access modifier
// Getter
public String getName() {
return name;
}
// Setter
public void setName(String newName) {
this.name = newName;
} }
```

Inheritance

Inheritance can be defined as the process where one class acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order.

```
class Subclass-name extends Superclass-name
{
//methods and fields
}
```

Interface / Abstraction:

It is use to provide privacy. It allows us to get a+b without showing that how it is shwoing

```
public interface network {
   abstract void call();
}
 class jazz implements inter{
@Override
public void call(){
    System.out.println("jazz call");
}public static void main(String[] args) {
        jazz obj=new jazz();
       obj.call();
    } }
```

Polymorphism (overloading/ over riding)

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

```
// A class with multiple methods with the same name
public class Adder {
// method 1
public void add(int a, int b) {
System.out.println(a + b);
}
// method 2
public void add(int a, int b, int c) {
System.out.println(a + b + c);
}
// method 3
public void add(String a, String b) {
System.out.println(a + " + " + b);
} }
// My main class
class MyMainClass {
public static void main(String[] args) {
Adder adder = new Adder(); // create a Adder object
adder.add(5, 4); // invoke method 1
adder.add(5, 4, 3); // invoke method 2
adder.add("5", "4"); // invoke method 3
}}
```

File Operations/ File handling

File handling refers to reading or writing data from files. Java provides some functions that allow us to manipulate data in the files.

File creation:

```
Import java.io;
File file=new File("filename.txt");
```

To create file we have to call createNewFile() function of file object.

To write something in the file

```
import FileWriter; // Import the FileWriter class
FileWriter fw = new FileWriter("filename.txt");
fw.write("Hi Wrting in file now ");
fw.close();
```

Reading data from file:

```
  Scanner fi = new Scanner(file);
while (  fi.hasNextLine()  ) {
   String data = fi.nextLine();
   System.out.println(data);
    }
```

File object Method	Use
createNewFile file.createNewFile()	It creates an empty file. And check whether file created or not. After making the file Class object, we have to run this method **to create file**.
getName file.getName()	It returns the name of the file

getAbsolutePath file.getAbsolutePath()	It returns absolute path name of file
canRead file.canRead()	Checks whether the file is readable or not
canWrite file.canWrite()	Checks whether the file is writable or not
exists file.exists()	Checks whether the file exists
length file.length()	It returns the size of the file in bytes
mkdir file.mkdir()	To create new directory
close file.close()	It is used to close the file
delete file.delete()	It deletes a file
setReadOnly file.setReadOnly()	It will make file to read only, for writing we have to change it's mode.
setWriteable file.setWritable(true);	It will make the file writeable.

Exception Handling

An exception is an unusual condition that results in an

interruption in the flow of the program.

try-catch-final block

try statement allow you to define a block of code to be tested for errors.
catch block is used to handle the exception.

finally code is executed whether an exception is handled or not.

```
try {
//Statements
}
catch (ExceptionType1 e1) {
// catch block
}
finally {
// finally block always executes
}
```