edureka!

JAVA OOP CHEAT SHEET

Learn **JAVA** from experts at https://www.edureka.co

Object Oriented Programming in Java

Java is an Object Oriented Programming language that produces software for multiple platforms. An object-based application in Java is concerned with declaring classes, creating objects from them and interacting between these objects.

Java Class

```
class Test {
        // class body
        member variables
        methods
 }
```

Java Object

```
//Declaring and Initializing an object
Test t = new Test();
```

Constructors

Default Constructor

```
class Test{
 /* Added by the Java Compiler at the Run Time
 public Test(){
 }
 */
 public static void main(String args[]) {
   Test testObj = new Test();
  }
}
```

Parameterized Constructor

```
public class Test {
   int appId;
   String appName;
//parameterized constructor with two parameters
Test(int id, String name){
   this.appId = id;
   this.appName = name;
}
void info(){
   System.out.println("Id: "+appId+" Name: "+appName);
}

public static void main(String args[]){
   Test obj1 = new Test(11001,"Facebook");
   Test obj2 = new Test(23003,"Instagram");
   obj1.info();
   obj2.info();
 }
}
```

JAVA CERTIFICATION TRAINING

Inheritance

Single Inheritance

```
Class A {
  //your parent class code
}
Class B extends A {
   //your child class code
}
```

Multi Level Inheritance

```
Class A {
  //your parent class code
}
Class B extends A {
   //your code
}
Class C extends B {
    //your code
}
```

Hybrid Inheritance

```
      A                          A
     / \                         |
    B   C          (OR)         B
   /     \                     / \
  D       E                   C   D
```

Hierarchical Inheritance

```
Class A {
  //your parent class code
}

Class B extends A {
   //your child class code
}

Class C extends A {
    //your child class code
}
```

Multiple Inheritance

```
Class A {
  //your parent class code
}
Class B {
   //your parent class code
}
Class C extends A,B {
    //your child class code
}
```

Polymorphism

Compile Time Polymorphism

```
class Calculator {
  static int add(int a, int b){
  return a+b;
 }
static double add( double a, double b){
  return a+b;
 }
public static void main(String args[]){
  System.out.println(Calculator.add(123,17));
  System.out.println(Calculator.add(18.3,1.9));
 }
}
```

Run Time Polymorphism

```
public class Mobile{
void sms(){System.out.println("Mobile class");}
}

//Extending the Mobile class
public class OnePlus extends Mobile{
//Overriding sms() of Mobile class
 void sms(){
  System.out.println(" OnePlus class");
 }

public static void main(String[] args) {
  OnePlus smsObj= new OnePlus();
  smsObj.sms();
 }
}
```

Abstraction

Abstract Class

```
public abstract class MyAbstractClass
 {
    public abstract void abstractMethod();
    public void display(){
     System.out.println("Concrete method");
  }
 }
```

Interface

```
//Creating an Interface
public interface Bike { public void start(); }
//Creating classes to implement Bike interface
class Honda implements Bike{
  public void start() {
   System.out.println("Honda Bike");
} }
class Apache implements Bike{
  public void start() {
   System.out.println("Apache Bike");
} }
class Rider{
  public static void main(String args[]){
   Bike b1=new Honda();
   b1.start();
   Bike b2=new Apache();
   b2.start();
} }
```

Encapsulation

```
public class Artist {
   private String name;
   //getter method
   public String getName() { return name; }
   //setter method
   public void setName(String name) { this.name = name; }
  }
public class Show{
  public static void main(String[] args){
   //creating instance of the encapsulated class
   Artist s=new Artist();
   //setting value in the name member
   s.setName("BTS");
   //getting value of the name member
   System.out.println(s.getName());
 }
}
```

Modifiers in Java

Access Modifiers

Scope	Private	Default	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Non - Access Modifiers

Type	Scope
Static	Makes the attribute dependent on a class
Final	Once defined, doesn't allow any changes
Abstract	Makes the classes and methods abstract
Synchronized	Used to synchronize the threads

Python

TunnelsUp.com v1.1

General Information

```
Whitespace matters! Indent where needed.
Import modules with "import modulename"
# This is a comment
print("Hello, World!") # prints to screen
```

Conditional Statements

```
if isSunny:
  print('It's sunny!')
elif 90 <= temp < 100 and bath > 80:
  print('Bath is hot and full!')
elif not ((job == 'qa') or (usr == 'adm')):
  print('Match if not qa or adm')
else:
  print('No match. Job is ' + job)
```

Lists

```
scores = ['A', 'C', 90, 75, 'C']
scores[0]          # 'A'
scores[1:3]        # 'C', 90
scores[2:]         # 90, 75, 'C'
scores[:1]         # 'A'
scores[:-1]        # 'A', 'C', 90, 75
len(scores)        # 5
scores.count('C')  # 2
scores.sort()      # 75, 90, 'A', 'C', 'C'
scores.remove('A') # removes 'A'
scores.append(100) # Adds 100 to list
scores.pop()       # removes the last item
scores.pop(2)      # removes the third item
75 in scores       # True
```

Numbers

```
total = 3 * 3          # 9
total = 5 + 2 * 3      # 11
cost = 1.50 + 3.75     # 5.25
total = int("9") + 1   # 10
```

Strings

```
title = 'Us and them'
# most list operations work on strings
title[0]           # 'U'
len(title)         # 11
title.split(' ')   # ['Us', 'and', 'them']
':'.join(['A','B','C']) # 'A:B:C'
nine = str(9)      # convert int to string
title.replace('them', 'us') # Us and us
```

Tuples

```
Like lists, except they cannot be changed
tuple1 = (1,2,3,"a","z")  # Creates tuple
tuple1[3]                 # 'a'
```

Dictionaries

```
votes = {'red': 3, 'blue': 5}
votes.keys()           # ['blue', 'red']
votes['gold'] = 4      # add a key/val
del votes['gold']      # deletes key
votes['blue'] = 6      # change value
len(votes)             # 2
votes.values()         # [6, 3]
'green' in votes       # False
votes.has_key('red')   # True
```

For Loops

```
grades = ['A', 'C', 'B', 'F']
for grade in grades:          # iterate over all vals
  print (grade)

for k,v in enumerate(grades): # using key value pair
  if v=='F':
      grades[k]='A'           # change all Fs to As

inv = {'apples': 7, 'peaches': 4}
for fruit, count in inv.items(): # using dictionaries
    print("We have {} {}".format(count, fruit))

for i in range(10):       # 0 to 9 counting by 1s
for i in range(5, 10):    # 5 to 9 counting by 1s
for i in range(9, 2, -1): # 9 to 3 decreasing by 1s
```

Functions

```
def sumNums(numOne, numTwo = 0):
  return numOne + numTwo

print(sumNums(3,4))          # 7
print(sumNums(3))            # 3
```

While Loops

```
i = 0
while True:
 i += 1
 if i == 3:
  continue # go to next loop
 if i == 7:
  break    # end loop
 print(i)  # 1 2 4 5 6
```

Class

```
class Person:
 def __init__(self, name, age):
  self.name = name
  self.age = age

 def birthYear(self):
  return year - self.age

user = Person('Jimmi', 27)
user.name = 'Jim'
print(user.name)   # prints Jim
print(user.birthYear())
```

</dream.in.code>
Programming & Web Development Community

C++ Reference Sheet

Include Headers
```
#include <headerfile>
```

Common Headers
iostream, fstream, math, cctype, string

Namespace
```
using namespace std;
```

Data Types
int, char, float, double, void, bool

Comments
```
// Comment text
/* Multi-line comment text */
```

Arithmetic Operators
*+ (Addition), - (Subtraction), * (Multiplication), / (Division), % (Modulus)*

Relational Operators
< (Less Than), <= (Less Than or Equal To),> (Greater Than), >= (Greather Than or Equal To),== (Equal To),!= (Not Equal To)

Logical Operators
|| (logical OR), && (logical AND), ! (logical NOT)

Pointers
```
int *ptr;   //Define pointer
ptr = &var  //ptr set to address of var
var2 = *ptr  //Set var2, to value of var1
```

If Else
```
if(<condition>)
   {  <statement 1>; }
else
   {  <statement 2>; }
```

For Loop
```
for(<initialize>;<condition>;<update>)
{  <statement>;  }
```

While Loop
```
while (<condition>)
{  <statement>;  }
```

Do-While Loop
```
do {  <statement>;  }
while (<condition>);
```

Switch Statement
```
switch(<expression> )
{
case <constant1>:
  <statement sequence 1>;
  break;
case <constant2>:
  <statement sequence 2>;
  break;

case <constantn+1>:
  <statement sequence n+1>;
  break;
[ default:
  <statement sequence n>;
  break;]
}
```

Arrays
```
//New 5 element array
int myArray[5];
//Array index starts at 0
//Access 3rd Element
myArray[2]=var;
```

I/O Operators
```
>> //Input Operator
<< //Output Operator
cin >> var1, var2, var3;
cout << "TEXT: " << var1 << endl;
cin.get(char* buffer, streamsize num, char delim );
```

File I/O
```
fstream  file;
file.open("filename",<file mode constant>);
   //Reads and Writes like cin and cout
      file >> var;
      file << "Text: "<< var << endl;
   // Read Entire Line
      getline (file,line);
   //Reading Writing Binary Data
      file.read(memory_block, size);
      file.write(memory_block, size);
file.close();
```

File Mode Constants
```
ios::in  //Opens file for reading
ios::out  //Opens file for writing
ios::ate  //Seeks the EOF.I/O operations can occur anywhere
ios::app  //Causes output to be appended at EOF
ios::trunc  //Destroys the previous contents
ios::nocreate  //Causes open() to fail if file doesnt already exist
ios::noreplace  //Causes open() to fail if file already exists
```

Function Prototype
```
<return_data_type> <function_name> (parameter list)
{ body of the function }
```

Class Prototype
```
class <class_name>
{
public:
   //method_prototypes
protected:
   //method_prototypes
private:
   //method_prototypes
   //data_attributes
};
```

Structure Prototype
```
struct <structure_name> {
member_type1 member_name1;
member_type2 member_name2;
} <object_name>;
```

Accessing Data Structures
```
//Access member variable from Struct/Class
myStruct.membervar1 = var;
//Call Class Method
myClass.method1(args);
//Pointer to Struct/Class
myStructType *ptr;
ptr = &myStruct;
ptr->membervar1 = var;
```


Edited By: frog, Amadeus, Splutard, Nova Dragoon, Dark_Nexus
Published: August 2, 2006