Frequently Asked Java Programs for QA

Top 10 numbers Questions

1. Swap two numbers

Input: a = 100, b= 200;
Output: a = 200, b= 100;

```
public static void main(String[] args) {
int a = 100, b = 200;
System.out.println("After swapping, a = " + a + " and b = " + b);
// 1. Swapping using three Variables
int temp = a;
a = b;
b = temp;
System.out.println("After swapping, a = " + a + " and b = " + b);
// 2. Using Two Variables
a = a + b;
b = a - b;
a = a - b;
System.out.println("After swapping, a = " + a + " and b = " + b);
// 3. Swapping a and b using XOR
a = a ^ b;
b = a ^ b;
a = a ^ b;
System.out.println("After swapping, a = " + a + " and b = " + b);
}
```

---

2. Armstrong number -

Armstrong number is a number that is equal to the sum of cubes of its digits.

Input: 153 , **Output:** Yes
153 is an Armstrong number. ==> (1*1*1) + (5*5*5) + (3*3*3) = 153

```
public static void main(String[] args) {
int sum = 0, res, temp;
int num = 153;// It is the number to check Armstrong
temp = num;
while (num > 0) {
res = num % 10;
num = num / 10;
sum = sum + (res * res * res);
}
if (temp == sum)
System.out.println(temp + " is armstrong number");
else
System.out.println(temp + " is Not armstrong number");
}
```

---

3. Fibonacci Series –

In Fibonacci series, next number is the sum of previous two numbers
Input = First 10 Numbers
Output = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 etc.
The first two numbers of Fibonacci series are 0 and 1.

```
public static void main(String[] args) {
int num1 = 0, num2 = 1, num=10;
for (int i = 0; i <= num; i++) {
System.out.print(num1 + " ");
int num3 = num2 + num1;// Swap
num1 = num2;
num2 = num3;
}
}
```

---

4. Reverse a numbers and Number is Palindrome or Not.

Input = 12321
Output =12321

```
public static void main(String[] args) {
int num = 12321;
// 1. Reverse a Number Using the While Loop reversed number
int rev = 0;
int temp = num;
int rem; // remainder
while (num > 0) {
rem = num % 10;
rev = (rev * 10) + rem;
num = num / 10;
}
System.out.println("Reversed Number is " + rev);
// Verify number is palindrome or not
if (rev == temp) {
System.out.println("palindrome number ");
} else {
System.out.println("not palindrome");
}
}
```

---

5. Factorial Number

Factorial Program in Java: Factorial of n is the product of all positive descending integers.

Input = 5!
Output = 5! = 5*4*3*2*1 = 120

```
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter number which you want for Factorial: ");
int num = sc.nextInt();
int fact = 1;
for (int i = 1; i <= num; i++) {
fact = fact * i;
}
System.out.println("Factorial of" + num + " is " + fact);
}
```

---

6. OddEvenNumbers

Input = 11
Output = Given number is odd number

```
public static void main(String[] args) {
// 1. Using Brute Forcew Approach
Scanner sc = new Scanner(System.in);
System.out.println("Enter Number:-");
int num = sc.nextInt();
if (num % 2 == 0)// Brute Forcew Approach
{
System.out.println("Given is even number");
} else {
System.out.println("Given number is odd number");
}
```

---

7. Prime Number

Prime number is a number that is greater than 1 and divided by 1 or itself only.

Input = 31, Output = The number is prime.

```
public static void main(String[] args) {
int num = 31;
int count = 0;
if (num <= 1) {
System.out.println("The number is not prime");
return;
}
for (int i = 2; i <= num / 2; i++) {
if (num % i == 0)
count++;
}
if (count > 1) {
System.out.println("The number is not prime");
} else {
System.out.println("The number is prime"); }
```

---

8. Largest number from 3 number/ given list

```
public static void main(String[] args) {
// TODO Auto-generated method stub
// 1. By using if else condition
int num1 = 7, num2 = 9, num3 = 10;
if( num1 >= num2 && num1 >= num3)
System.out.println(num1 + " is the largest number.");
else if (num2 >= num1 && num2 >= num3)
System.out.println(num2 + " is the largest number.");
else
System.out.println(num3 + " is the largest number.");

// 2. Using Collections.max() method and ArrayList
ArrayList<Integer> x = new ArrayList<>();
x.add(12);
x.add(22);
x.add(54);
System.out.println(Collections.max(x)+ " is the largest number.");
}
```

---

9. Sum of Digits

Sum of all given numbers.

Input = 987

Output = 24

```
public static void main(String[] args) {
int n = 987;
int sum = 0;
while (n != 0) {
sum = sum + n % 10;
n = n / 10;
}
System.out.println("Using While:- " + sum);
}
```

---

10. Count digits in an integer number

Input = 29845315, **Output = 8**

```
public static void main(String[] args) {
// TODO Auto-generated method stub
long num = 29845315;
int count = 0, num2 = 298453;
// 1. by using while loop
while (num != 0) {
```

```
num = num / 10;
count++;
}
System.out.println("Number of digits : " + count);
// 2. Converting given number to string solution to count digits in an integer
String result = Integer.toString(num2); // calculate the size of string
System.out.println(+result.length());
}
```

Top 15 String Questions

1. Reverse a string

Input = mama
Output = mama

```
public static void main(String[] args) {
String str = "mama";
String s2 = "";
// 1. by using the charAt() method
    for (int i = str.length() - 1; i >= 0; i--) {
    s2 = s2 + str.charAt(i);// extracts each character and store in string
    }
System.out.println("Reversed word: " + s2);
// below is code to check weather given string is Palindrome or not
if (str.equalsIgnoreCase(s2)) {
System.out.println("String is Palindrome");
} else {
System.out.println("String is not Palindrome");
}
}
// 2. Using built in reverse() method of the StringBuilder class:
String input = "Welcome To Jave Learning";
StringBuilder input1 = new StringBuilder();
input1.append(input); // append a string into StringBuilder input1
input1.reverse();
System.out.println(input1);

// 3. Using StringBuffer:
String strText = "Java Learning";
// conversion from String object to StringBuffer
StringBuffer sbr = new StringBuffer(strText);
sbr.reverse();
System.out.println(sbr);
```

---

2. Remove space form given string

Input String = “hello java   Learning ”
Output String = “hellojavaLearning”

```
public static void main(String[] args) {
System.out.println("Enter String ");
Scanner sc = new Scanner(System.in);
String input = sc.nextLine();
System.out.println("Original String- " + input);
input = input.replaceAll("\\s", "");
System.out.println("Final String- " + input);
}
```

---

3. Finding Common Elements in Arrays

Input =
array1 = { 4, 2, 3, 1, 6 }; array2 = { 6, 7, 8, 4 };
Output = 6,4

```
// By using the for loop

Integer[] array1 = { 4, 2, 3, 1, 6 };
Integer[] array2 = { 6, 7, 8, 4 };
List<Integer> commonElements = new ArrayList<>();
for (int i = 0; i < array1.length; i++) {
for (int j = 0; j < array2.length; j++) {
if (array1[i] == array2[j]) {
commonElements.add(array1[i]);
} }}
System.out.println("Common Elements for given two array List is:" +
commonElements);

// by using ArrayList with retainAll method
ArrayList<Integer> list1 = new ArrayList<>(Arrays.asList(array1));
ArrayList<Integer> list2 = new ArrayList<>(Arrays.asList(array2));
list1.retainAll(list2);
System.out.println("Common Elements:" + list1);

// By using Java Stream
String[] array3 = { "Java", "JavaScript", "C", "C++" };
String[] array4 = { "Python", "C#", "Java", "C++" };
ArrayList<String> list3 = new ArrayList<>(Arrays.asList(array3));
ArrayList<String> list4 = new ArrayList<>(Arrays.asList(array4));
List<String> commonElements1 =
list3.stream().filter(list4::contains).collect(Collectors.toList());
System.out.println(commonElements1);
} }
```

---

4. Find first and last element of ArrayList in java

Input = array1 = { 4, 2, 3, 1, 6 };
Output = First is:4, Last is: 6

```
ArrayList<Integer> list = new ArrayList<Integer>(5);
// find first element
 int first = list.get(0);//First Element
 // find last element
 int last = list.get(list.size() - 1);//last Element
```

---

5. Second Largest and Second Smallest Numbers:

```
// Code to find second largest and second smallest numbers in an array
int[] arrayList = { 4, 2, 3, 1,0, 6,12,15,20 };
int num=arrayList.length;
Arrays.sort(arrayList);
System.out.println("Second Largest element is "+arrayList[num-2]); //Display Second Smallest
System.out.println("Second Smallest element is "+arrayList[1]);
```

---

6. How to sort an Array without using inbuilt method?

Input = array[] = { 10, 5, 20, 63, 12, 57, 88, 60 };

Output = 5 10 12 20 57 60 63 88

```
int temp, size;
int array[] = { 10, 5, 20, 63, 12, 57, 88, 60 };
size = array.length;
for (int i = 0; i < size; i++) {
for (int j = i + 1; j < size; j++) {
if (array[i] > array[j]) {
temp = array[i];
array[i] = array[j];
array[j] = temp;
}}}
for (int i = 0; i < array.length; i++) {
System.out.println("Array sorted: " + array[i]);
}
// Print 3rd Largest number from an Array
System.out.println("Third largest number is:: " + array[size - 3]);
System.out.println("**********************");

// sort array using the Arrays.sort method
Arrays.sort(array);
System.out.println("sorted array- " + Arrays.toString(array));
int thirdMaxNum=array[size-3];
System.out.println("Third highest array- " +thirdMaxNum );
```

---

7. Counting number of occurrences of given word in a string using Java?

String = "Java is a programming language. Java is widely used in software Testing";

Input = ”Java”, **Output** = 2

```
public static void main(String[] args) {
String string = "Java is a programming language. Java is widely used in software
Testing";
String[] words = string.toLowerCase().split(" ");
String word = "java";
int occurrences = 0;
for (int i = 0; i < words.length; i++)
if (words[i].equals(word))
occurrences++;
System.out.println(occurrences);
}
```

---

8. Find each word occurrence from given string in string java

Input = "Alice is girl and Bob is boy";
Output = {Bob=1, Alice=1, and=1, is=2, girl=1, boy=1}

```
public static void main(String[] args) {
String str = "Alice is girl and Bob is boy";
Map<String, Integer> hashMap = new HashMap<>();
String[] words = str.split(" ");
for (String word : words) {
if (hashMap.containsKey(word))
hashMap.put(word, hashMap.get(word) + 1);
else
hashMap.put(word, 1);
}
System.out.println(hashMap);
```

---

9. Reverse the entire sentence

Input = "India is country My"

Output = "My country is India"

```
public static void main(String[] args) {
String str[] = "India is country My".split(" ");
String ans = "";
for (int i = str.length - 1; i >= 0; i--) {
ans = ans + str[i] + " ";
}
System.out.println(ans.substring(0, ans.length() - 1));
}
```

10. count the occurrences of each character?

Input = "This is an example";

Output = p = 1, a = 2, s = 2, T = 1, e = 2, h = 1, x = 1, i = 2, l = 1, m = 1, n = 1

```
public static void main(String[] args) {
String str = "This is an example";
HashMap<Character, Integer> count = new HashMap<Character, Integer>();
// convert string to character array
char[] arr = str.toCharArray();
// traverse every character and count the Occurrences
for (char c : arr) {
// filter out white spaces
if (c != ' ') {
if (count.containsKey(c)) {
// if character already traversed, increment it
count.put(c, count.get(c) + 1);
} else {
// if character not traversed, add it to hashmap
count.put(c, 1);
}
}
}
// traverse the map and print the number of occurences of a character
for (Map.Entry entry : count.entrySet()) {
System.out.print( entry.getKey() + " = " + entry.getValue()+", ");
}
}
```

---

11. Removing Duplicates from an Array

```
// using for loop
String[] strArray = {"abc", "def", "abc", "mno", "xyz", "pqr", "xyz", "pqr"};
//1. Using Brute Force Method
for (int i = 0; i < strArray.length-1; i++)
{
for (int j = i+1; j < strArray.length; j++)
{
if( (strArray[i]==(strArray[j])) )
{
System.out.println("Brute Force Method : Duplicate Element is : "+strArray[j]);
}}}
// using Hashset
HashSet<String> hs = new HashSet<String>();
for (String arrayElement : strArray)
{
if(!hs.add(arrayElement))
{System.out.println("HashSet :Duplicate Element is : "+arrayElement);
}}
```

---

12. Reverse each word in a sentence

Input = "reverse each word in a string";

Output = "esrever hcae drow ni a gnirts"

```
public static void main(String[] args) {
String str = "reverse each word in a string";
String words[] = str.split("\\s");
String reverseWord = "";
for (String w : words) {
StringBuilder sb = new StringBuilder(w);
sb.reverse();
reverseWord = reverseWord + sb.toString() + " ";
}
System.out.println(reverseWord.trim());
}
```

---

13. String Anagrams: Determine if two strings are anagrams of each other

Input =
String str1 = "Army";
String str2 = "Mary";
Output = **army** and **mary** are anagram.

```
public static void main(String[] args) {
String str1 = "Army";
String str2 = "Mary";
str1 = str1.toLowerCase();
str2 = str2.toLowerCase();
// check if length is same
if (str1.length() == str2.length()) {
// convert strings to char array
char[] charArray1 = str1.toCharArray();
char[] charArray2 = str2.toCharArray();
// sort the char array
Arrays.sort(charArray1);
Arrays.sort(charArray2);
// if sorted char arrays are same, then the string is anagram
boolean result = Arrays.equals(charArray1, charArray2);
if (result) {
System.out.println(str1 + " and " + str2 + " are anagram.");
} else {
System.out.println(str1 + " and " + str2 + " are not anagram.");
}
} else {
System.out.println(str1 + " and " + str2 + " are not anagram.");
}
```

---

14. How to print duplicate characters from the string?

Input = "apple is fruit";

Output = p i

```
public static void main(String[] args) {
String str = "apple is fruit";
char[] carray = str.toCharArray();
System.out.println("The string is:" + str);
System.out.print("Duplicate Characters in above string are: ");
for (int i = 0; i < str.length(); i++) {
for (int j = i + 1; j < str.length(); j++) {
if (carray[i] == carray[j]) {
System.out.print(carray[j] + "");
break;
}
}
}
}
```

---

15. Find and print the largest element in an array.

```
// Initialize array
int[] arr = new int[] { 25, 11, 7, 75, 56 };
// Initialize max with first element of array.
int max = arr[0];
// Loop through the array
for (int i = 0; i < arr.length; i++) {
// Compare elements of array with max
if (arr[i] > max)
max = arr[i];
}
System.out.println("Largest element present in given array: " + max);
```

---

16. Java program to split an alphanumeric digit without using split method

Input = "Welcome234To567Java89Programming0@#!!";

Output =

WelcomeToJavaProgramming

234567890

@#!!

```
public static void main(String[] args) {
String str = "Welcome234To567Java89Programming0@#!!";
StringBuffer alpha = new StringBuffer(), num = new StringBuffer(), special = new
StringBuffer();
for (int i = 0; i < str.length(); i++) {
```

```
if (Character.isDigit(str.charAt(i)))
num.append(str.charAt(i));
else if (Character.isAlphabetic(str.charAt(i)))
alpha.append(str.charAt(i));
else
special.append(str.charAt(i));
}
System.out.println(alpha);
System.out.println(num);
System.out.println(special);
}
```

Python cheat sheet

(COMMONLY USED CODE SNIPPETS)

1. Basic Python Syntax:

Task	Code
Print to Console	print("Hello, World!")
Variable Assignment	x = 10
Commenting	# This is a comment
Multi-line Comment	''' This is a multi-line comment '''
Input from User	name = input("Enter your name: ")
Check Data Type	type(x)
Type Casting	int("10"), float("10.5"), str(100)

2. Data Structures:

Task	Code
List (Array)	my_list = [1, 2, 3, 4, 5]
Access List Item	my_list[0]
List Slicing	my_list[1:4]
Add Item to List	my_list.append(6)
Remove Item from List	my_list.remove(3)
Tuple	my_tuple = (1, 2, 3, 4)
Set	my_set = {1, 2, 3, 4}
Dictionary (HashMap)	my_dict = {"key1": "value1", "key2": "value2"}
Access Dictionary Value	my_dict["key1"]
Add Key-Value Pair	my_dict["key3"] = "value3"

3. Control Flow:

Task	Code
If Statement	if x > 10: print("x is greater than 10")
If-Else Statement	if x > 10: print("x is greater than 10") else: print("x is less than or equal to 10")
Elif Statement	if x > 10: print("x is greater") elif x == 10: print("x is 10") else: print("x is smaller")
For Loop	for i in range(5): print(i)
While Loop	while x < 10: x += 1
Break	for i in range(5): if i == 3: break
Continue	for i in range(5): if i == 3: continue

4. Functions:

Task	Code
Define Function	def my_function(): print("Hello from function!")
Function with Parameters	def greet(name): print(f"Hello, {name}!")
Return Value from Function	def add(a, b): return a + b
Lambda Function	add = lambda a, b: a + b

5. String Manipulation:

Task	Code
Concatenate Strings	full_name = "John" + " " + "Doe"
String Length	len("Hello")
Convert to Upper Case	"hello".upper()
Convert to Lower Case	"HELLO".lower()
Substring	"Hello, World!"[7:12]
Find Substring	"Hello, World!".find("World")
Replace Substring	"Hello, World!".replace("World", "Python")
Split String	"Hello, World!".split(",")

6. File Handling:

Task	Code
Open a File	file = open("example.txt", "r")
Read File	content = file.read()
Read Line by Line	lines = file.readlines()
Write to a File	file = open("example.txt", "w"); file.write("Hello, World!")
Close a File	file.close()

7. List Comprehension:

Task	Code
Basic List Comprehension	[x**2 for x in range(5)]
List Comprehension with Condition	[x for x in range(10) if x % 2 == 0]

8.Error Handling:

Task	Code
Try-Except Block	try: x = 10 / 0 except ZeroDivisionError: print("Cannot divide by zero")
Finally Block	try: x = 10 / 0 except ZeroDivisionError: print("Error!") finally: print("This runs always")

9. Working with Libraries:

Task	Code
Importing a Library	import math
Using a Library Function	math.sqrt(16)
Install a Library (using pip)	pip install pandas
Import Specific Function	from math import sqrt

10. NumPy for Numerical Operations:

Task	Code
Import NumPy	import numpy as np
Create NumPy Array	arr = np.array([1, 2, 3, 4, 5])
Array Reshaping	arr.reshape(5, 1)
Array Operations	arr + 10, arr * 2
Array Slicing	arr[1:4]
Array Statistics	np.mean(arr), np.median(arr), np.std(arr)

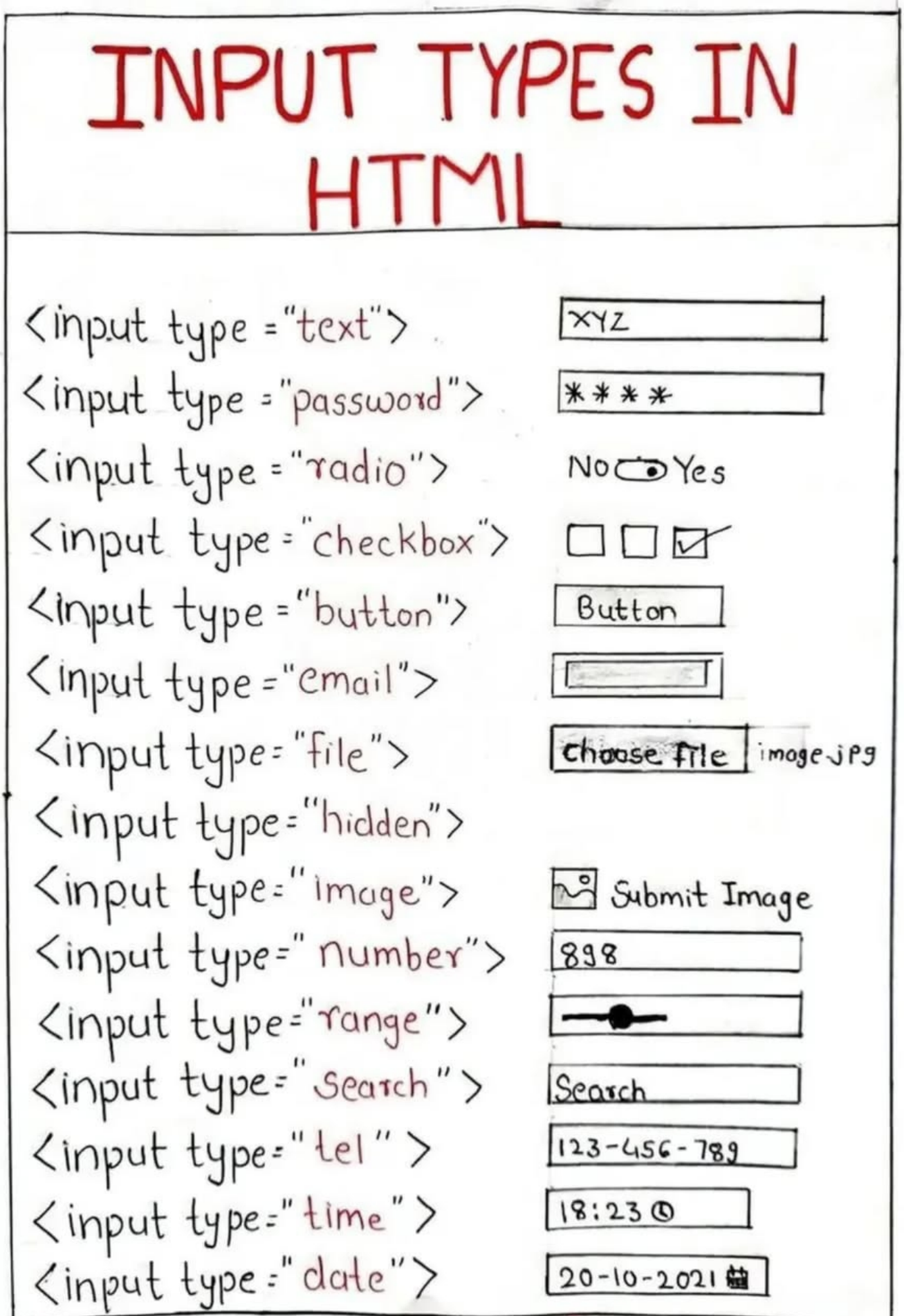
11. Pandas for Data Handling:

Task	Code
Import Pandas	import pandas as pd
Create DataFrame	df = pd.DataFrame({"Name": ["Alice", "Bob"], "Age": [25, 30]})
Read CSV File	df = pd.read_csv("data.csv")
View Data	df.head()
Basic Statistics	df.describe()
Filter Data	df[df["Age"] > 25]
Group By	df.groupby("Age").mean()

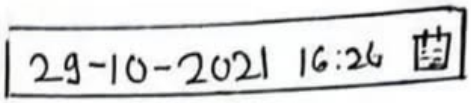
12. Matplotlib for plotting:

Task	Code
Import Matplotlib	import matplotlib.pyplot as plt
Simple Plot	plt.plot([1, 2, 3], [4, 5, 6]); plt.show()
Bar Plot	plt.bar([1, 2, 3], [4, 5, 6]); plt.show()
Histogram	plt.hist([1, 2, 2, 3, 4, 5]); plt.show()
Scatter Plot	plt.scatter([1, 2, 3], [4, 5, 6]); plt.show()

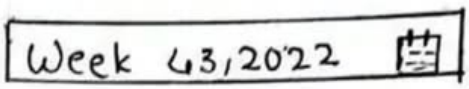
INPUT TYPES IN HTML



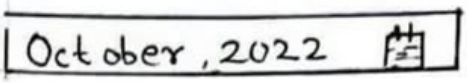
<input type="datetime-local">



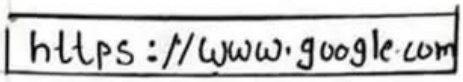
<input type="week">



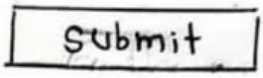
<input type="month">



<input type="url">

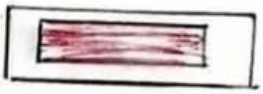


<input type="submit">



<input type="reset">

<input type="color">



Follow for More

Such amazing content

Part 2

Top 50 OOPs Interview Questions & Answers

Here are OOPs interview questions and answers for fresher as well experienced candidates to get their dream job.

1) What is OOPS?

OOPS is abbreviated as Object Oriented Programming system in which programs are considered as a collection of objects. Each object is nothing but an instance of a class.

2) Write basic concepts of OOPS?

Following are the concepts of OOPS:

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

3) What is a class?

A class is simply a representation of a type of object. It is the blueprint/plan/template that describes the details of an object.

---

4) What is an Object?

An object is an instance of a class. It has its own state, behavior, and identity.

---

5) What is Encapsulation?

Encapsulation is an attribute of an object, and it contains all data which is hidden. That hidden data can be restricted to the members of that class.

Levels are Public, Protected, Private, Internal, and Protected Internal.

6) What is Polymorphism?

Polymorphism is nothing but assigning behavior or value in a subclass to something that was already declared in the main class. Simply, polymorphism takes more than one form.

---

7) What is Inheritance?

Inheritance is a concept where one class shares the structure and behavior defined in another class. If Inheritance applied to one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple Inheritance.

---

8) What are manipulators?

Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

---

9) Explain the term constructor

A constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules for constructor are:

- Constructor Name should be the same as a class name.
- A constructor must have no return type.

---

10) Define Destructor?

A destructor is a method which is automatically called when the object is made of scope or destroyed. Destructor name is also same as class name but with the tilde symbol before the name.

---

11) What is an Inline function?

An inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

---

12) What is a virtual function?

A virtual function is a member function of a class, and its functionality can be overridden in its derived class. This function can be implemented by using a keyword called virtual, and it can be given during function declaration.

A virtual function can be declared using a token(virtual) in C++. It can be achieved in C/Python Language by using function pointers or pointers to function.

---

13) What is a friend function?

A friend function is a friend of a class that is allowed to access to Public, private, or protected data in that same class. If the function is defined outside the class cannot access such information.

A friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public, or protected.

---

14) What is function overloading?

Function overloading is a regular function, but it is assigned with multiple parameters. It allows the creation of several methods with the same name which differ from each other by the type of input and output of the function.

Example

```
void add(int& a, int& b);

void add(double& a, double& b);

void add(struct bob& a, struct bob& b);
```

---

15) What is operator overloading?

Operator overloading is a function where different operators are applied and depends on the arguments. Operator,-,* can be used to pass through the function, and it has its own precedence to execute

---

16) What is an abstract class?

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with an abstract class, but it can be inherited. An abstract class can contain only an Abstract method. Java allows only abstract method in abstract class while other languages allow non-abstract method as well.

---

17) What is a ternary operator?

The ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types, and it depends on the function. The ternary operator is also called a conditional operator.

---

18) What is the use of finalize method?

Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected, and it is accessible only through this class or by a derived class.

---

19) What are the different types of arguments?

A parameter is a variable used during the declaration of the function or subroutine, and arguments are passed to the function body, and it should match with the parameter defined. There are two types of Arguments.

- Call by Value – Value passed will get modified only inside the function, and it returns the same value whatever it is passed into the function.
- Call by Reference – Value passed will get modified in both inside and outside the functions and it returns the same or different value.

---

20) What is the super keyword?

The super keyword is used to invoke the overridden method, which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass.

It also forwards a call from a constructor, to a constructor in the superclass.

---

21) What is method overriding?

Method overriding is a feature that allows a subclass to provide the implementation of a method that overrides in the main class. It will override the implementation in the superclass by providing the same method name, same parameter, and same return type.

---

22) What is an interface?

An interface is a collection of an abstract method. If the class implements an interface, it thereby inherits all the abstract methods of an interface.

Java uses Interface to implement multiple inheritances.

---

23) What is exception handling?

An exception is an event that occurs during the execution of a program. Exceptions can be of any type – Runtime exception, Error exceptions. Those exceptions are adequately handled through exception handling mechanism like try, catch, and throw keywords.

---

24) What are tokens?

A compiler recognizes a token, and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals, and operators are examples of tokens.

Even punctuation characters are also considered as tokens. Example: Brackets, Commas, Braces, and Parentheses.

---

25) What is the main difference between overloading and overriding?

Overloading is static Binding, whereas Overriding is dynamic Binding. Overloading is nothing but the same method with different arguments, and it may or may not return the equal value in the same class itself.

Overriding is the same method names with the same arguments and return types associated with the class and its child class.

---

26) What is the main difference between a class and an object?

An object is an instance of a class. Objects hold multiple information, but classes don't have any information. Definition of properties and functions can be done in class and can be used by the object.

A class can have sub-classes, while an object doesn't have sub-objects.

---

27) What is an abstraction?

Abstraction is a useful feature of OOPS, and it shows only the necessary details to the client of an object. Meaning, it shows only required details for an object, not the inner constructors, of an object. Example – When you want to switch on the television, it is not necessary to know the inner circuitry/mechanism needed to switch on the TV. Whatever is required to switch on TV will be shown by using an abstract class.

---

28) What are the access modifiers?

Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are five types of access modifiers, and they are as follows:

- Private
- Protected
- Public
- Friend
- Protected Friend

---

29) What are sealed modifiers?

Sealed modifiers are the access modifiers where the methods can not inherit it. Sealed modifiers can also be applied to properties, events, and methods. This modifier cannot be used to static members.

---

30) How can we call the base method without creating an instance?

Yes, it is possible to call the base method without creating an instance. And that method should be "Static method."

Doing Inheritance from that class.-Use Base Keyword from a derived class.

---

31) What is the difference between new and override?

The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.

---

32) What are the various types of constructors?

There are three types of constructors:

– Default Constructor – With no parameters.

– Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.

– Copy Constructor – Which creates a new object as a copy of an existing object.

---

33) What is early and late Binding?

Early binding refers to the assignment of values to variables during design time, whereas late Binding refers to the assignment of values to variables during run time.

---

34) What is 'this' pointer?

THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. It refers to the current object.

---

35) What is the difference between structure and a class?

The default access type of a Structure is public, but class access type is private. A structure is used for grouping data, whereas a class can be used for grouping data and methods. Structures are exclusively used for data, and it doesn't require strict validation, but classes are used to encapsulate and inherent data, which requires strict validation.

---

36) What is the default access modifier in a class?

The default access modifier of a class is Internal and the default access modifier of a class member is Private.

---

37) What is a pure virtual function?

A pure virtual function is a function which can be overridden in the derived class but cannot be defined. A virtual function can be declared as Pure by using the operator =0.

Example –

```
Virtual void function1() // Virtual, Not pure
```

Virtual void function2() = 0 //Pure virtual

---

38) What are all the operators that cannot be overloaded?

Following are the operators that cannot be overloaded -.

1. Scope Resolution (::)
2. Member Selection (.)
3. Member selection through a pointer to function (.*)

---

39) What is dynamic or run time polymorphism?

Dynamic or Run time polymorphism is also known as method overriding in which call to an overridden function is resolved during run time, not at the compile time. It means having two or more methods with the same name, same signature but with different implementation.

---

40) Do we require a parameter for constructors?

No, we do not require a parameter for constructors.

---

41) What is a copy constructor?

This is a special constructor for creating a new object as a copy of an existing object. There will always be only one copy constructor that can be either defined by the user or the system.

---

42) What does the keyword virtual represented in the method definition?

It means we can override the method.

---

43) Whether static method can use nonstatic members?

False.

---

44) What are a base class, subclass, and superclass?

The base class is the most generalized class, and it is said to be a root class. A Subclass is a class that inherits from one or more base classes.

The superclass is the parent class from which another class inherits.

---

45) What is static and dynamic Binding?

Binding is nothing but the association of a name with the class. Static Binding is a binding in which name can be associated with the class during compilation time, and it is also called as early Binding.

Dynamic Binding is a binding in which name can be associated with the class during execution time, and it is also called as Late Binding.

---

46) How many instances can be created for an abstract class?

Zero instances will be created for an abstract class. In other words, you cannot create an instance of an Abstract Class.

---

47) Which keyword can be used for overloading?

Operator keyword is used for overloading.

---

48) What is the default access specifier in a class definition?

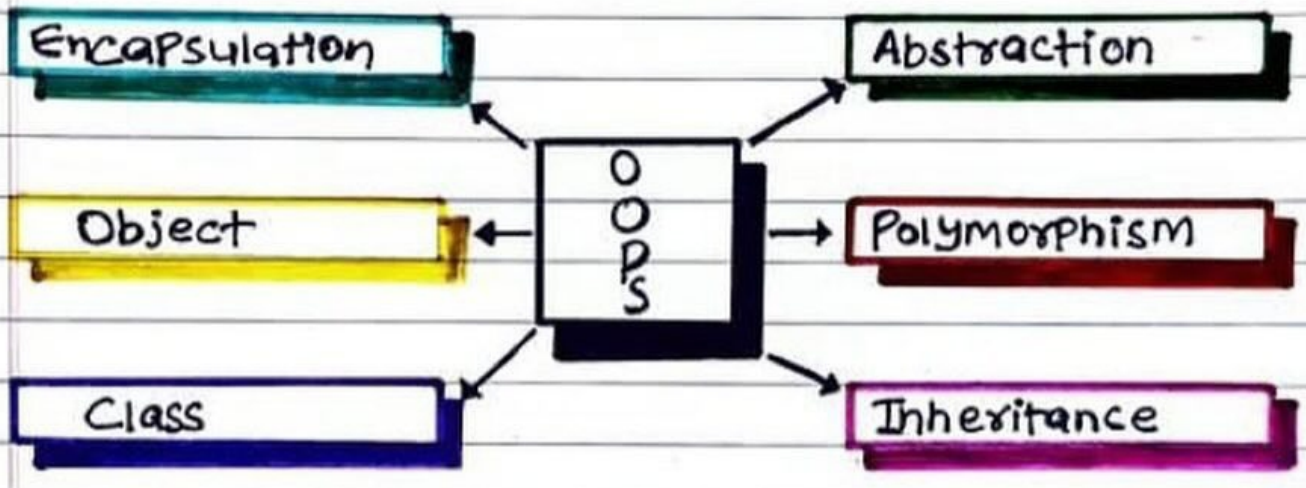
Private access specifier is used in a class definition.

OOPS in C++

The Main aim of OOP is to bind together the data and the functions the operate on them so that no other part of the code can access this data except this function.

ATUL KUMAR (LINKEDIN).
NOTES GALLERY (TELEGRAM).

TYPES OF OOPS.



⇨ **Class :** It is a user defined data types, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.

⇨ **Object :** When a class is defined no memory is allocated but when it is instantiated (i.e., Object is created) memory is allocated.

⇨ **Encapsulation :** In OOP, Encapsulation is defined as binding together the data and the functions that manipulates them.

⇨ **Abstraction :** Abstraction means displaying only essential information and hiding the details.

- Abstraction using classes
- Abstraction using Header files (math.h → Pow())

⇨ **Polymorphism :** In simple words, we can define polymorphism as an ability of a message to be displayed in more than one form.

- Operator overloading
- function overloading
  ↳ int sum(10, 20, 30)
  int sum(10, 20)

⇨ **Inheritance :** The capability of a class to derive properties and characteristics from another class is called inheritance.

- Subclass
- SuperClass
- Reusability

Dynamic Binding :

In dynamic binding, the code to be executed in response to function call is decided at run time.

Constructors :

A constructor is a member function of a class which initializes objects of a class. In C++ constructor is automatically called when the object creates.
It has same name as class itself.
Constructor don't have a return type.

1. Default constructor (NO parameter passed)
2. Parametrized Constructor
3. Copy constructor.

Destructor in C++ :

Derived class destructor will be invoked first, then the base class destructor will be invoked.

Access Modifier :

Public :- can be accessed by any class.
Private :- can be accessed only by a function in a class (inaccessible outside the class).
Protected :- It is also inaccessible outside the class but can be accessed by subclass at that class.

Note :- If we do not specify any access modifier inside the class then by default the access modifier for the member will be private.

Friend class :

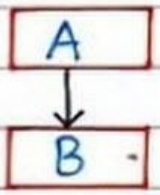
A Friend class can access private and protected members of other class in which it is declared as friend.

Ex :- friend class B;

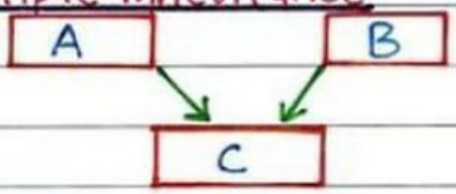
- **Inheritance**

```
Class Subclass : accessmode . baseclass
{ ——
}
```

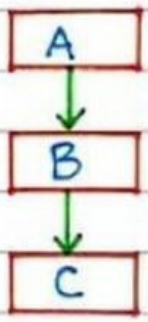
1. **Single inheritance :**



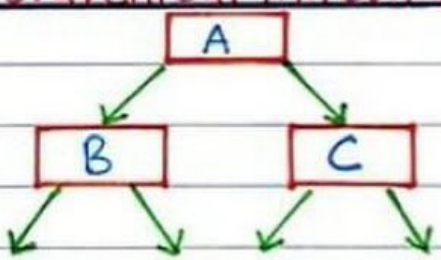
2. **Multiple inheritance**



3. **Multilevel**



4. **Hierarchical inheritance**



5. **Hybrid**
Combination of one or more type.

- **Polymorphism**
→ Compile time Poly → Operator overloading.
→ Function overloading.

→ Runtime Poly

↳ Function overriding occurs when a derive class has a definition of one or more members of base class.

Advantages of Data Abstraction

- Avoid code duplication and inc. reusability.
- Can change internal implementation of class independently.

Structure Vs Class

Most important difference is security.
A Structure is not secure and cannot hide its member function and variable while class is secure and can hide its programming & designing details.

Local classes in C++ :

A class declared inside a function becomes local to that function and is called local class.
All the methods of local class must be defined inside the class only.

Virtual function and Runtime Polymorphism :

⇒ A virtual function is a member function which is declared with a base class and redefined (overriden) by derived class. functions are declared with virtual keyword base class.

⇨ Runtime Polymorphism, also known as the Dynamic Method Dispatch, is a process that resolves a call to an overridden method at runtime. The process involves the use of the reference variable of a superclass to call for an overridden method.

Exception Handling in C++ :

try: represent a block of code that can throw an exception.

catch: represent a block of code that get executed when error is thrown.

throw: Used to throw an exception.

There is a special catch block → catch(...)
It catches all types of error.

- ## Inline Function

Inline is a request not command.
It is function that is expanded in line when it is called. When the inline function is called, whole code get inserted or substituted at the point of inline function call.

```
inline return-type function( )
{ ==
}
```

Function Overloading

Function overloading is a feature in C++ where two or more functions can have same name but different parameters.

```
Void print (int i )
{     cout<< "Here is int " <<i <<endl;
}
Void print (float i )
{     cout << "Here is float" <<i<< endl;
}
 int main
{        print (10) ;
         print (10.12) ;
}
```

Differences b/w C and C++

C	C++
• C supports procedural program.	• C++ is known as hybrid language, because it support both procedural and object orlented Programming.
• As C does not support the oops concept so it has no support for polymorphism, encapsulation and inheritance.	• C++ has support for Polymorphism, encapsulation and inheritance as it is an OOPS language.
• C is a subset of C++	• C++ is a superset of C.

• C contains 32 Keywords	• C++ Contain 52 Keywords (public, private, protected, try, catch, throw)
• C is a function driven language.	• C++ is an object driven language.
• Function and operator overloading is not support in C.	• C++ supports function & operator overloading.
• C does not support exception handling.	• C++ supports exception handling using try and catch.

- Structure is a collection of dissimilar elements.

- Static Members in C++
  - Static variable in a Function :

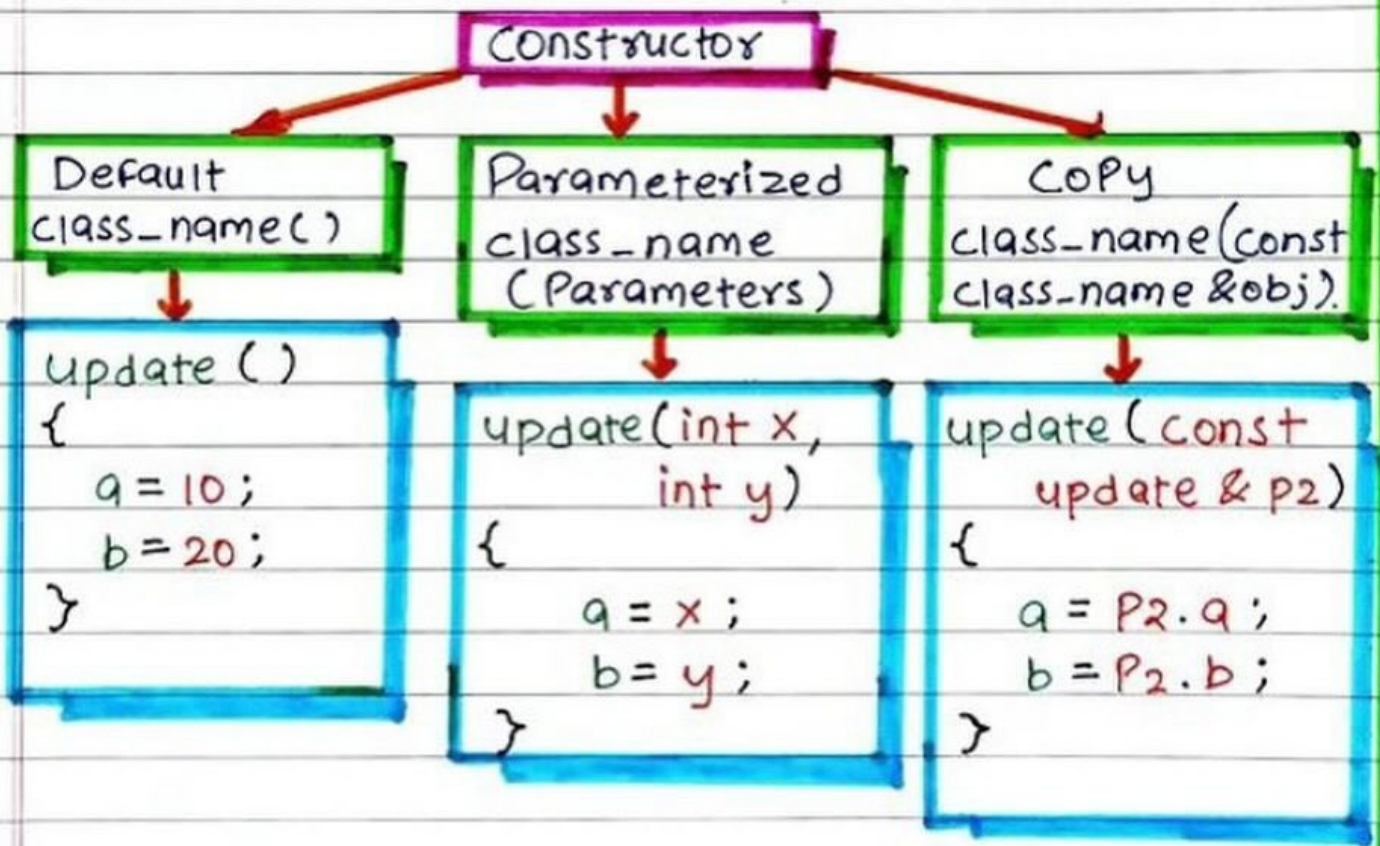
When a variable is declared as static, space for it gets allocated for the lifetime of the program. (default intialized to 0)
Even if the function is called multiple times, the space for it is allocated once.

- Static variable in a class :

> Declared inside the class body.
> Also known as class member variable.
> They must be defined outside the class.
> Static variable doesn't belong to any object, but to the whole class.

Constructors :

- Constructors is an special member function of the class. It is automatically invoked when an object is created.
- It has no return type.
- Constructor has same name as class itself.
- If we do not specify, then C++ compiler generates a default constructor for us.



Compiler generates two constructor by itself.

1. Default Constructor
2. Copy Constructor

But if any of the constructor is created by user, then default constructor will not be created by compiler.

#_ Comprehensive Python CheatSheet

1. Basic Operations

- Print to console: `print("Hello, World!")`
- Get user input: `name = input("Enter your name: ")`
- String concatenation: `full_name = first_name + " " + last_name`
- String formatting (f-string): `print(f"Hello, {name}!")`
- String formatting (.format): `print("Hello, {}!".format(name))`
- String formatting (%): `print("Hello, %s!" % name)`
- Type conversion (to int): `age = int("25")`
- Type conversion (to float): `price = float("19.99")`
- Type conversion (to string): `age_str = str(25)`
- Check type: `type(variable)`
- Get length: `len(sequence)`
- Get memory address: `id(object)`
- Delete variable: `del variable_name`
- Check if object is instance of class: `isinstance(object, class)`
- Get all attributes of an object: `dir(object)`

2. Numbers and Math

- Addition: `result = 5 + 3`
- Subtraction: `result = 10 - 4`
- Multiplication: `result = 6 * 7`
- Division: `result = 15 / 3`
- Integer division: `result = 17 // 3`
- Modulus: `remainder = 17 % 3`
- Exponentiation: `result = 2 ** 3`
- Absolute value: `abs(-5)`
- Round number: `round(3.7)`
- Round to specific decimal places: `round(3.14159, 2)`
- Floor division: `import math; math.floor(5.7)`
- Ceiling division: `import math; math.ceil(5.2)`
- Square root: `import math; math.sqrt(16)`
- Calculate pi: `import math; math.pi`
- Calculate e: `import math; math.e`
- Logarithm (base e): `import math; math.log(10)`
- Logarithm (base 10): `import math; math.log10(100)`
- Sine: `import math; math.sin(math.pi/2)`

- Cosine: `import math; math.cos(math.pi)`
- Tangent: `import math; math.tan(math.pi/4)`
- Degrees to radians: `import math; math.radians(180)`
- Radians to degrees: `import math; math.degrees(math.pi)`
- Generate random number: `import random; random.random()`
- Generate random integer: `import random; random.randint(1, 10)`
- Choose random element: `import random; random.choice([1, 2, 3, 4, 5])`

3. Strings

- Create string: `s = "Hello, World!"`
- Multiline string: `s = """This is a multiline string"""`
- Raw string: `s = r"C:\Users\John"`
- String repetition: `"Hello" * 3`
- String indexing: `first_char = s[0]`
- String slicing: `substring = s[1:5]`
- Reverse string: `reversed_string = s[::-1]`
- Convert to uppercase: `s.upper()`
- Convert to lowercase: `s.lower()`
- Capitalize string: `s.capitalize()`
- Title case: `s.title()`
- Swap case: `s.swapcase()`
- Strip whitespace: `s.strip()`
- Left strip: `s.lstrip()`
- Right strip: `s.rstrip()`
- Replace substring: `s.replace("old", "new")`
- Split string: `parts = s.split(",")`
- Join strings: `",".join(["a", "b", "c"])`
- Check if string starts with: `s.startswith("Hello")`
- Check if string ends with: `s.endswith("World!")`
- Find substring: `index = s.find("World")`
- Count occurrences: `count = s.count("l")`
- Check if string is alphanumeric: `s.isalnum()`
- Check if string is alphabetic: `s.isalpha()`
- Check if string is digit: `s.isdigit()`
- Check if string is lowercase: `s.islower()`
- Check if string is uppercase: `s.isupper()`
- Check if string is title case: `s.istitle()`
- Check if string is whitespace: `s.isspace()`

4. Lists

- Create list: `lst = [1, 2, 3, 4, 5]`
- Create list with range: `lst = list(range(1, 6))`
- Access element: `element = lst[0]`
- Slice list: `subset = lst[1:4]`
- Append to list: `lst.append(6)`
- Extend list: `lst.extend([7, 8, 9])`
- Insert at index: `lst.insert(0, 0)`
- Remove by value: `lst.remove(3)`
- Remove by index: `lst.pop(2)`
- Clear list: `lst.clear()`
- Index of element: `index = lst.index(4)`
- Count occurrences: `count = lst.count(2)`
- Sort list: `lst.sort()`
- Sort list in reverse: `lst.sort(reverse=True)`
- Reverse list: `lst.reverse()`
- Copy list: `new_lst = lst.copy()`
- Shallow copy: `import copy; new_lst = copy.copy(lst)`
- Deep copy: `import copy; new_lst = copy.deepcopy(lst)`
- List comprehension: `squares = [x**2 for x in range(10)]`
- Filter with list comprehension: `evens = [x for x in range(10) if x % 2 == 0]`
- Nested list comprehension: `matrix = [[i*j for j in range(5)] for i in range(5)]`
- Flatten nested list: `flattened = [item for sublist in nested_list for item in sublist]`
- Zip lists: `zipped = list(zip(list1, list2))`
- Unzip lists: `unzipped = list(zip(*zipped))`
- Check if element in list: `5 in lst`
- Get max value: `max(lst)`
- Get min value: `min(lst)`
- Sum of list: `sum(lst)`
- Join list of strings: `" ".join(lst)`
- Create list of lists: `matrix = [[0 for _ in range(5)] for _ in range(5)]`

5. Tuples

- Create tuple with single element: `t = (1,)`
- Access element: `element = t[0]`
- Slice tuple: `subset = t[1:3]`
- Concatenate tuples: `new_t = t + (4, 5, 6)`
- Repeat tuple: `repeated_t = t * 3`
- Count occurrences: `count = t.count(2)`
- Index of element: `index = t.index(3)`
- Check if element in tuple: `2 in t`
- Unpack tuple: `a, b, c = t`
- Swap values using tuple: `a, b = b, a`
- Convert list to tuple: `t = tuple([1, 2, 3])`
- Convert tuple to list: `lst = list(t)`
- Create tuple of tuples: `matrix = ((1, 2, 3), (4, 5, 6), (7, 8, 9))`
- Named tuple: `from collections import namedtuple; Point = namedtuple('Point', ['x', 'y']); p = Point(1, 2)`

6. Sets

- Create set: `s = {1, 2, 3}`
- Create set from list: `s = set([1, 2, 3, 3, 2, 1])`
- Add element: `s.add(4)`
- Update set: `s.update([4, 5, 6])`
- Remove element: `s.remove(2)`
- Remove element if present: `s.discard(5)`
- Pop random element: `element = s.pop()`
- Clear set: `s.clear()`
- Union of sets: `union = s1 | s2`
- Intersection of sets: `intersection = s1 & s2`
- Difference of sets: `difference = s1 - s2`
- Symmetric difference: `sym_diff = s1 ^ s2`
- Check if subset: `is_subset = s1.issubset(s2)`
- Check if superset: `is_superset = s1.issuperset(s2)`
- Check if disjoint: `is_disjoint = s1.isdisjoint(s2)`
- Frozen set (immutable): `fs = frozenset([1, 2, 3])`

7. Dictionaries

- Create dictionary: `d = {"key": "value"}`
- Create dictionary with dict(): `d = dict(key="value")`
- Access value: `value = d["key"]`

- Access value with default: `value = d.get("key", "default")`
- Add/update key-value pair: `d["new_key"] = "new_value"`
- Update dictionary: `d.update({"key1": "value1", "key2": "value2"})`
- Remove key-value pair: `del d["key"]`
- Remove and return value: `value = d.pop("key")`
- Remove and return last item: `item = d.popitem()`
- Get keys: `keys = d.keys()`
- Get values: `values = d.values()`
- Get key-value pairs: `items = d.items()`
- Clear dictionary: `d.clear()`
- Copy dictionary: `new_d = d.copy()`
- Deep copy dictionary: `import copy; new_d = copy.deepcopy(d)`
- Check if key in dictionary: `"key" in d`
- Dictionary comprehension: `squares = {x: x**2 for x in range(5)}`
- Merge dictionaries (Python 3.5+): `merged = {**dict1, **dict2}`
- Get value of nested dictionary: `value = d['outer_key']['inner_key']`
- Default dictionary: `from collections import defaultdict; dd = defaultdict(int)`
- Ordered dictionary: `from collections import OrderedDict; od = OrderedDict()`
- Counter dictionary: `from collections import Counter; c = Counter(['a', 'b', 'c', 'a', 'b', 'a'])`

8. Control Flow

- If statement: `if condition: do_something()`
- If-else statement: `if condition: do_something() else: do_other_thing()`
- If-elif-else statement: `if condition1: do_something() elif condition2: do_other_thing() else: do_default()`
- Ternary operator: `result = x if condition else y`
- For loop: `for item in iterable: do_something()`
- For loop with index: `for index, item in enumerate(iterable): do_something()`
- For loop with range: `for i in range(5): do_something()`
- While loop: `while condition: do_something()`
- Break from loop: `if condition: break`
- Continue to next iteration: `if condition: continue`
- Else clause in for loop: `for item in iterable: do_something() else: no_break_occurred()`

- Else clause in while loop: `while condition: do_something() else: condition_is_false()`
- Pass statement: `if condition: pass`
- Match-case (Python 3.10+): `match value: case pattern: do_something()`
- Loop over multiple lists: `for item1, item2 in zip(list1, list2): do_something()`
- Nested loops: `for i in range(3): for j in range(3): print(i, j)`
- List comprehension with if: `[x for x in range(10) if x % 2 == 0]`
- Dictionary comprehension with if: `{x: x**2 for x in range(5) if x % 2 == 0}`

9. Functions

- Define function: `def func_name(param): return result`
- Function with default parameter: `def greet(name="World"): print(f"Hello, {name}!")`
- Function with multiple parameters: `def func(param1, param2, param3): pass`
- Function with variable arguments: `def sum_all(*args): return sum(args)`
- Function with keyword arguments: `def print_info(**kwargs): for k, v in kwargs.items(): print(f"{k}: {v}")`
- Lambda function: `square = lambda x: x**2`
- Return multiple values: `def func(): return 1, 2, 3`
- Nested function: `def outer(): def inner(): pass; return inner`
- Closure: `def outer(x): def inner(y): return x + y; return inner`
- Decorator: `def decorator(func): def wrapper(*args, **kwargs): return func(*args, **kwargs); return wrapper`
- Apply decorator: `@decorator def function(): pass`
- Partial function: `from functools import partial; add_five = partial(add, 5)`
- Recursive function: `def factorial(n): return 1 if n == 0 else n * factorial(n-1)`
- Generator function: `def gen(): yield item`
- Asynchronous function: `async def async_func(): await asyncio.sleep(1)`

10. Classes and Object-Oriented Programming

- Define class: `class ClassName: pass`
- Create instance: `obj = ClassName()`
- Define constructor: `def __init__(self, param): self.param = param`

- Define method: `def method_name(self): pass`
- Define class method: `@classmethod def class_method(cls): pass`
- Define static method: `@staticmethod def static_method(): pass`
- Inheritance: `class ChildClass(ParentClass): pass`
- Multiple inheritance: `class ChildClass(Parent1, Parent2): pass`
- Call superclass method: `super().method_name()`
- Property decorator: `@property def prop_name(self): return self._prop`
- Setter decorator: `@prop_name.setter def prop_name(self, value): self._prop = value`
- Abstract base class: `from abc import ABC, abstractmethod; class AbstractClass(ABC): @abstractmethod def abstract_method(self): pass`
- Dataclass (Python 3.7+): `from dataclasses import dataclass; @dataclass class Point: x: float; y: float`
- Method overriding: `def method_name(self): # Override parent method`
- Private attribute: `self.__private_attr = value`
- Name mangling: `obj._ClassName__private_attr`
- Duck typing: `if hasattr(obj, 'method_name'): obj.method_name()`
- Context manager class: `class ContextManager: def __enter__(self): pass; def __exit__(self, exc_type, exc_value, traceback): pass`
- Metaclass: `class Meta(type): pass; class MyClass(metaclass=Meta): pass`

11. Exceptions and Error Handling

- Try-except block: `try: do_something() except Exception as e: handle_error(e)`
- Try-except-else block: `try: result = do_something() except Exception: handle_error() else: use_result(result)`
- Try-except-finally block: `try: do_something() except Exception: handle_error() finally: cleanup()`
- Catch multiple exceptions: `try: do_something() except (TypeError, ValueError) as e: handle_error(e)`
- Raise exception: `raise ValueError("Invalid input")`
- Raise from: `raise ValueError("Invalid input") from original_error`
- Assert statement: `assert condition, "Error message"`
- Custom exception: `class CustomError(Exception): pass`
- Handle all exceptions: `try: do_something() except Exception as e: handle_any_error(e)`
- Re-raise exception: `try: do_something() except Exception as e:`

- Exception chaining: `try: do_something() except Exception as e: raise RuntimeError("Operation failed") from e`

12. File I/O and Context Managers

- Open file: `with open("file.txt", "r") as f: content = f.read()`
- Write to file: `with open("file.txt", "w") as f: f.write("content")`
- Append to file: `with open("file.txt", "a") as f: f.write("new content")`
- Read lines from file: `with open("file.txt", "r") as f: lines = f.readlines()`
- Write lines to file: `with open("file.txt", "w") as f: f.writelines(lines)`
- Read file line by line: `with open("file.txt", "r") as f: for line in f: print(line)`
- Check if file exists: `import os; os.path.exists("file.txt")`
- Get file size: `import os; os.path.getsize("file.txt")`
- Delete file: `import os; os.remove("file.txt")`
- Rename file: `import os; os.rename("old_name.txt", "new_name.txt")`
- Create directory: `import os; os.mkdir("new_directory")`
- Change directory: `import os; os.chdir("path/to/directory")`
- Get current working directory: `import os; cwd = os.getcwd()`
- List directory contents: `import os; files = os.listdir("directory")`
- Walk directory tree: `import os; for root, dirs, files in os.walk("directory"): print(root, dirs, files)`

13. Modules and Packages

- Import module: `import module_name`
- Import specific item: `from module_name import item_name`
- Import with alias: `import module_name as alias`
- Import all items: `from module_name import *`
- Reload module: `import importlib; importlib.reload(module_name)`
- Get module attributes: `dir(module_name)`
- Get module help: `help(module_name)`
- Create package: `# Create __init__.py in directory`
- Relative import: `from .module import item`
- Absolute import: `from package.module import item`
- Import from parent directory: `from ..module import item`
- Check if module is main: `if __name__ == "__main__": main()`