Explore More

Subcription : Premium CDAC NOTES & MATERIAL @99

Contact to Join
Premium Group

Click to Join
Telegram Group

For More E-Notes

Join Our Community to stay Updated

TAP ON THE ICONS TO JOIN!

	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql
51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9clHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

Java Stream API Interview Questions & Answers (MCQs)

Q#1. What is the output of the following code?

```
List<Integer> numbers = Arrays.asList(11, 20, 33, 45, 52);
int sum = numbers.stream()
                 .filter(n -> n % 4 == 0)
                 .map(n -> n * 2)
                 .reduce(0, Integer::sum);
System.out.println(sum);
```

(A) 72
(B) 100
(C) 144
(D) 2052

Ans: C. The code filters out the numbers divisible by 4 (20 and 52), doubles them (40 and 104), and then adds them together (144). The reduce() method starts with an initial value of 0 and applies the Integer::sum function to each element of the stream to compute the final sum of 144.

Q#2. Which of the following methods can be used to convert a stream of strings to a list of integers?

(A) mapToInt(Integer::parseInt)
(B) map(Integer::parseInt)
(C) flatMapToInt(Integer::parseInt)
(D) mapToInt(String::valueOf)

Ans: A. The mapToInt() method is used to convert a stream of objects to an IntStream, which has additional methods for integer-specific operations like summation, max, and min. The Integer::parseInt method is passed as the mapping function to convert each string to an integer.

Q#3. What is the output of the following code?

```
Stream.of(1, 2, 3)
       .flatMap(n -> Stream.of(n, n * 2))
       .forEach(System.out::print);
```

(A) 123246
(B) 122436
(C) 123123
(D) 246246

Ans: B. In this case, the function passed to flatMap() creates a new stream with the original integer and the integer multiplied by 2. So, for example, the integer 1 will be mapped to a stream containing 1 and 2, the integer 2 will be mapped to a stream containing 2 and 4, and so on. Finally, the forEach() method is used to print each element in the resulting stream to the console using the method reference System.out::print. The result is the concatenated sequence of integers: 122436.

Q#4. What is the output of the following code?

```
List<Integer> numbers = Arrays.asList(12, 8, 13, 24, 15);

Optional<Integer> result = numbers.stream()
                                  .filter(n -> n > 25)
                                  .reduce((a, b) -> a + b);
System.out.println(result);
```

(A) 0
(B) null
(C) Optional.empty
(D) Compilation error

Ans: C. The code creates a stream from the numbers list, and uses the filter() operation to filter any numbers that are greater than 25. Since there are no numbers left in the stream after the filter, the reduce() operation has nothing to work with and returns an Optional.empty value.

Q#5. What is the output of the following code?

```
List<String> words = Arrays.asList("hello", "world");
List<Character> letters = words.stream()
                               .flatMap(s -> Stream.of(s.split("")))
                               .collect(Collectors.toList());
System.out.println(letters);
```

(A) [hello, world]
(B) [h, e, l, l, o, w, o, r, l, d]
(C) [h, w, e, o, l, r, l, d, l]
(D) [h, e, llo, w, o, rld]

Ans: **B.** The flatMap() operation is applied to each String in the stream. The flatMap() method takes a function that returns another stream and flattens the resulting streams into a single stream. In this case, the function passed to flatMap() applies the split() method to each String, which splits the String into an array of its individual characters. Finally, the resulting List of Strings is printed to the console using the System.out.println() method. The result is a List containing each character of the original strings as a separate String element.

Q#6. What is the output of the following code?

```
IntStream.rangeClosed(1, 10)
         .filter(n -> n % 2 == 0)
         .forEach(System.out::print);
```

(A) 13579
(B) 246810

```
(C) 2468
(D) None of the Above
```

Ans. B. The code generates a stream of integers from 1 to 10 (inclusive), filters out the odd numbers, and then prints the even numbers using the forEach() method.

Q#7. Which of the following is an example of a stateful intermediate operation in Java 8 Stream?

```
(A) map()
(B) filter()
(C) sorted()
(D) distinct()
```


Ans: D. The distinct() operation removes duplicates from a stream, but it requires the entire stream to be buffered to ensure that no element is repeated. This makes it a stateful intermediate operation.

Q#8. What is the output of the following code if we try to print the result?

```
Stream.of("Cognizant", "Infosys", "Amdocs")
      .map(s -> s.length())
      .reduce(0, Integer::sum);
```

```
(A) 19
(B) 28
(C) 24
(D) 22
```

Ans: **D**. The code maps each string to its length, resulting in a stream of integers [9, 7, 6]. The reduce() method then computes the sum of the integers (22), starting with an initial value of 0.

Q#9. Which of the following methods can be used to convert a stream of integers to a list of strings?

```
(A) map(String::parseInt)
(B) mapToInt(Integer::parseInt)
(C) map(String::valueOf)
(D) mapToInt(String::valueOf)
```

Ans: **C**. The map() method is used to apply a function to each element of the stream, resulting in a stream of strings. The String::valueOf method is passed as the mapping function to convert each integer to its string representation.

Q#10. What is the output of this code?

```
Map<String, Integer> ages = new HashMap<>();
  ages.put("Robert", 30);
  ages.put("Mary", 25);
  ages.put("Peterson", 40);
  ages.put("Jinny", 35);


int result = ages.entrySet().stream()
                 .filter(entry -> entry.getValue() > 30)
                 .mapToInt(entry -> entry.getValue())
                 .sum();
System.out.println(result);
```

(A) 65
(B) 70
(C) 75
(D) 35

Ans: C. The code creates a HashMap of names and ages, and then creates a stream of Map.Entry objects using the entrySet() method. The filter() operation filters out the entries that have a value greater than 30. The mapToInt() operation is then used to convert the resulting entries to their values, and the sum() operation is used to compute the sum of all the values in the stream. The resulting sum is 40 + 35 = 75.

Q#11. Which of the following stream operations is used to generate an infinite stream of elements in Java 8?

(A) `stream()`
(B) `forEach()`
(C) `generate()`
(D) `parallelStream()`

Ans: C. The generate() method is used to generate an infinite stream of elements. It takes a Supplier as an argument, which produces new values to add to the stream.

Q#12. What is the output of the following code?

```
Stream.of("java", "javascript", "angular")
      .filter(s -> s.startsWith("b"))
      .findFirst()
      .ifPresent(System.out::println);
```

(A) java
(B) javascript
(C) angular
(D) Nothing is printed.

Ans: D. The code filters the stream to include only elements that start with the letter "b", finds the first element of the resulting stream, and prints it using ifPresent(). Since there is no element in the stream which starts with letter "b", the program will be terminated without printing anything.

Q#13. What is the output of the following code?

```
IntStream.range(0, 5)
         .mapToObj(b-> "a" + b)
         .sorted(Comparator.reverseOrder())
         .forEach(System.out::println);
```

(A) a1a2a3a4
(B) a4a3a2a1a0
(C) 1234
(D) 43210

Ans: **B.** The code generates a stream of integers from 0 to 4 (exclusive), maps each integer to a string starting with the letter "a", sorts the resulting strings in reverse order using sorted(), and then prints the strings using forEach().

Q#14. Which of the following stream methods is used to obtain a new stream consisting of the first n elements of the original stream?

(A) skip(n)
(B) limit(n)
(C) filter(n)
(D) map(n)

Ans: **B**. The limit(n) operation is used to obtain a new stream consisting of the first n elements of the original stream.

Q#15. What is the output of the following code?

```
Stream.of("java", "python", "react")
      .flatMap(s -> s.chars().boxed())
      .forEach(System.out::print);
```

(A) 106971189711212111610411111011410197991116
(B) jprjprjavapyhtonreact
(C) jpr
(D) javapythonreact

Ans: **A**. The code maps each string to a stream of characters, and then flattens the resulting stream of streams into a single stream of boxed integers using flatMap(). The resulting stream will contain the Unicode values of the characters in each string, which are then concatenated together into a single stream.The forEach() method then prints the elements of the stream.

Q#16. Which of the following stream methods is used to obtain a new stream consisting of the elements of the original stream that satisfy a given predicate?

(A) distinct()
(B) sorted()

```
(C) filter()
(D) map()
```

Ans: **C.** The filter() operation is used to obtain a new stream consisting of the elements of the original stream that accepts a specific predicate.

Q#17. Which of the following stream methods is used to combine the elements of the stream into a single result?

```
(A) reduce()
(B) collect()
(C) peek()
(D) forEach()
```

Ans: **A.** The reduce() method is used to combine the elements of the stream into a single result. It takes a binary operator as an argument that is used to combine the elements.

Q#18. What is the output of the following code?

```
Stream.of("spring", "hibernate", "jdbc")
      .map(s -> s.substring(2, 3))
      .forEach(System.out::print);
```



```
(A) pid
(B) pribdb
(C) rbb
(D) The code does not compile.
```

Ans: **C.** The code maps each string to a substring consisting of the third character of the each string using map(), and then prints the resulting substrings using forEach().

Q#19. Which of the following stream methods is used to obtain a new stream that contains only the elements of the original stream that are not null?

(A) filterNotNull()
(B) removeNull()
(C) filter()
(D) map()

Ans: **C**. The filter() operation can be used with a null check to obtain a new stream containing only the elements of the original stream that are not null.

Q#20. What is the output of the following code?

```
IntStream.rangeClosed(1, 5)
         .map(i -> i * i)
         .skip(2)
         .forEach(System.out::print);
```

(A) 1491625
(B) 91625
(C) 916
(D) 14916

Ans: **B**. The code generates a stream of integers from 1 to 5 (inclusive), maps each integer to its square using map() which results into [1,4,9,16,25], skips the first two squares using skip(), and then prints the resulting squares using forEach().

Q#21. What is the output of the following code?

```
Stream.of("A", "B", "C")
      .flatMap(s -> Stream.of(s, s.toLowerCase()))
      .forEach(System.out::print);
```

(A) ABCabc
(B) AaBbCc
(C) AbC
(D) The code does not compile.

Ans: **B**. The code applies flatMap() to each element of the stream, which generates a new stream containing the original element and its lowercase counterpart. The resulting stream of streams is then flattened to a single stream, which is printed using forEach().

Q#22. What is the output of the following code?

```
List<Integer> numbers = Arrays.asList(5, 4, 3, 2, 1);
int sum = numbers.stream()
                 .reduce(10, (a, b) -> a + b);
System.out.println(sum);
```

(A) 15
(B) 25
(C) 10
(D) Compilation Error in the code.

Ans: **B**. The code uses the reduce() operation to sum the integers in the numbers list and assigns the resulting sum to the sum variable. The sum is then printed as 25 because first argument in the reduce() method is already 10.

Q#23. What is the output of the following code?

```
Stream.of(3, 4, 2, 5, 1)
      .map(i -> i * 2)
      .skip(1)
```

```
        .limit(3)
        .forEach(System.out::print);
```

(A) 8410
(B) 16425
(C) 425
(D) 84102

Ans: A. The code first applies the map() operation to the stream to obtain a new stream of double of the original elements. It then skips the first element using skip() and limits the stream to the next three elements using limit(). Finally, it prints the resulting elements using forEach(). Therefore, the output is 8410.

Q#24. What is the output of the following code snippet?

```
List<String> words = Arrays.asList("java", "springboot", "angular", "javascript", "
String result = words.stream()
                     .filter(w -> w.length() > 15)
                     .findFirst()
                     .orElse("none");
System.out.println(result);
```


(a) none
(b) Washington
(c) New York
(d) An error is thrown

Ans: **A.** The code creates a list of strings, and then creates a stream from the list. The stream is filtered to only include strings with a length greater than 15, and the first element of the resulting stream is retrieved using the findFirst() method. If no element is found, the orElse() method returns the string “none”.

Q#25. Which of the following is not a terminal operation in Java Stream API?

(A) reduce()
(B) collect()
(C) filter()
(D) findFirst()

Ans: **C**. filter() is an intermediate operation in Java Stream API, which is used to filter out elements from a stream based on some condition. It returns a new stream that contains only the elements that satisfy the condition. The terminal operations in Java Stream API are the ones that produce a result or a side effect, such as reduce(), collect(), findFirst(), etc.

Q#26. Which of the following statements about the reduce() operation in Java Stream API is correct?

(A) The reduce() operation can only be used with numerical streams.
(B) The reduce() operation always returns a single value.
(C) The reduce() operation is a terminal operation.
(D) The reduce() operation is an intermediate operation.

Ans: **C.** The reduce() operation in Java Stream API is a terminal operation that combines the elements of a stream into a single value. It can be used with any type of stream, not just numerical streams. The reduce() operation returns an Optional object that may or may not contain the result, depending on whether the stream was empty or not.

Q#27. What is the output of the following code?

```
IntStream.range(24, 96)
         .limit(3)
         .mapToObj(Integer::toString)
         .forEach(System.out::print);
```

(A) 949596
(B) 252627

```
(C) 242526
(D) 939495
```

Ans: **C**. The code generates a stream of integers from 24 to 96 (exclusive), limits the stream to the first 3 elements, maps each integer to a string, and then prints the resulting strings using forEach().

Q#28. What is the output of this code?

```
List<String> words = Arrays.asList("one", "two", "three", "four", "five");
Map<Integer, List<String>> result = words.stream()
            .collect(Collectors.groupingBy(String::length));
System.out.println(result);
```

```
(A) {3=[one, two], 4=[four, five], 5=[three]}
(B) {one=3, two=3, three=5, four=4, five=4}
(C) {3=[one, two], 4=[four, five], 5=[three], null=[]}
(D) Compilation error in the code
```


Ans: **A**. The code creates a stream from the words list, and uses the groupingBy() collector from the Collectors class to group the elements of the stream into a Map where the keys are the lengths of the strings, and the values are lists of strings with that length. The resulting Map is printed to the console.

Q#29. What is the output of this code?

```
List<Integer> numbers = Arrays.asList(15, 24, 33, 12, 19);

Map<Boolean, List<Integer>> result = numbers.stream()
                .collect(Collectors.partitioningBy(n -> n % 2 == 0));

System.out.println(result);
```

```
(A) {even=[24, 12], odd=[15, 33, 19]}
(B) {1=[24, 12], 2=[15, 33, 19]}
(C) {false=[15, 33, 19], true=[24, 12]}
(D) Compilation error in the code
```

Ans: **C**. The code creates a stream from the numbers list, and uses the partitioningBy() collector from the Collectors class to partition the elements of the stream into a Map where the keys are booleans representing whether the elements satisfy the given predicate (in this case, whether they are even or odd), and the values are lists of elements that satisfy or don't satisfy the predicate. The resulting Map is printed to the console.

Q#30. What is the output of this code?

```
List<String> words = Arrays.asList("java", "typescript", "angular", "javascript", "
Set<String> result = words.stream()
                          .flatMap(s -> Arrays.stream(s.split("")))
                          .collect(Collectors.toSet());
System.out.println(result);
```



```
(A) [a, c, e, g, a, i, j, l, n, e, p, v, r, s, t, r, u, v, y]
(B) [a, c, e, g, i, j, l, n, p, r, s, t, u, v, y]
(C) [java, typescript, angular, javascript, react]
(D) [a, y, c, e, g, i, k, j, l, n, p, r, s, t, u, v, y]
```

Ans: **B**. The code creates a stream from the words list, and uses the flatMap() operation to transform each word into a stream of its individual characters. The resulting stream of characters is then collected into a Set (no repetition of characters) using the toSet() collector from the Collectors class. The resulting set is printed to the console.

Q#31. What is the output of this code?

```
List<String> words = Arrays.asList("one", "two", "three", "four", "five");
Map<Integer, String> result = words.stream()
```

```
                .collect(Collectors.toMap(String::length,
                    Function.identity(), (s1, s2) -> s1 + "|" + s2));
System.out.println(result);
```

(A) {3=one, 4=five, 5=three, 6=four}

(B) {3=one|two, 4=five|four, 5=three}

(C) {3=one, two, 4=five, four, 5=three}

(D) Compilation error

Ans: **B.** The code creates a stream from the words list, and uses the toMap() collector from the Collectors class to create a map with the length of each word as the key, and the word itself as the value. However, if multiple words have the same length, the toMap() collector uses a merge function to combine the values. In this case, the merge function concatenates the two values.

Q#1. Which of the following statement is incorrect about method overloading and method overriding?

- **A.** In method overloading, type of method parameters can be different, while in method overriding they must be same.
- **B.** In overloading, methods may have different return types, but in case of overriding methods must have same or covariant return types.
- **C.** Methods with modifiers 'private', 'static', and 'final' can be overloaded & overridden as well.
- **D.** Overriding is an example of Runtime polymorphism, while overloading is an example of compile-time polymorphism.

Answer: C
Explanation: Methods with modifiers 'private', 'static', and 'final' can be overloaded, but can't be overridden. JDK 1.5 onward, the overriding method's return type can be the sub-type of it's parent class method's return type. This is called **covariant return type.** The **covariant return type** always works only for non-primitive return types. Pleasse check an example of covariant return type. You may also check the complete detail on what is Method Overloading & Method Overriding?.

Q#2. Which of the following code snippets correctly demonstrates method overloading?

- **A)**

```
public void display(String message) { }
public int display(String message) { return 0; }
```

- **B)**

```
public void display(String message) { }
public void display(int number) { }
```

- **C)**

```
public int display(String message, int number) { return 0; }
public String display(String message, int number) { return number; }
```

- **D)**

```
public void display(String message) { }
public void display(String message, String additional) { }
```

Answer: B, D
Explanation: In method overloading, methods in the same class share the same name but have different parameter types or counts. Options B and D satisfy these conditions.

Q#3. Which OOP concept is used to derive a new class from an existing class, inheriting its attributes and behaviors?

- **A.** Composition
- **B.** Inheritance
- **C.** Encapsulation
- **D.** Polymorphism

Answer: B
Explanation: Inheritance allows one class to derive from another, inheriting attributes and behaviors, promoting code reuse and hierarchy.

Q#4. Select the code snippet from below options that best demonstrates the concept of abstraction.

- **A)**

```
abstract class Vehicle {
   abstract void startEngine();
}
```

- **B)**

```
class Vehicle {
   public void startEngine() {
        System.out.println("Engine started.");
   }
}
```

- **C)**

```
interface Vehicle {
    void startEngine();
}
```

- **D)**

```
class Car {
  private String model;
  void startEngine() { }
}
```

Answer: A, C
Explanation: Abstraction hides implementation details, which can be achieved with abstract classes and interfaces, as shown in options A and C.

Q#5. In a graphic editor, shapes like Circle, Square, and Triangle need to implement a common draw() method. Which concept allows each shape to implement its own version of draw() having common method signature?

- **A.** Inheritance
- **B.** Method Overloading
- **C.** Method Overriding
- **D.** Polymorphism

Answer: C, D
Explanation: Polymorphism and method overriding enable each shape to provide its specific implementation of draw() method, adhering to a common method signature.

Q#6. Which of the following code snippets correctly implements method overriding?

- **A)**

```
class Animal {
    void makeSound() {
       System.out.println("Animal sound");
    }
}
```

```
class Dog extends Animal {
    void makeSound() {
       System.out.println("Bark");
    }
}
```

- **B)**

```
class Animal {
    static void makeSound() {
       System.out.println("Animal sound");
    }
}

class Dog extends Animal {
    static void makeSound() {
       System.out.println("Bark");
    }
}
```

- **C)**

```
class Animal {
    void makeSound(String sound) {
       System.out.println(sound);
    }
}

class Dog extends Animal {
    void makeSound() {
       System.out.println("Bark");
    }
}
```

- **D)**

```
class Animal {
    void makeSound() {
        System.out.println("Animal sound");
    }
}

class Dog {
    void makeSound() {
        System.out.println("Bark");
    }
}
```

Answer: A
Explanation: In method overriding, the subclass provides a specific implementation for a method already defined in its superclass with the same signature, as in option A. Since static methods can't be overridden, therefore option B is incorrect. In option C method arguments don't match. Option D doesn't have inheritance relationship.

Q#7. What is the main purpose of a constructor in a class?

- **A.** To initialize objects of the class
- **B.** To inherit from another class
- **C.** To override methods from the superclass
- **D.** To call methods with specific parameters

Answer: A
Explanation: A constructor is known as a special method used to initialize objects when they are created, setting initial values or configurations as needed.

Q#8. You are designing a Library class that contains multiple Book objects, where the Book objects exist only within the Library. Which relationship best fits in this situation?

- **A.** Aggregation
- **B.** Association
- **C.** Composition
- **D.** Inheritance

Answer: C
Explanation: Composition implies a strong relationship where the lifecycle of Book objects is tied to the Library. When Library is destroyed, the Book objects are also destroyed.

Q#9. Which of the following code snippets demonstrates polymorphism?

- **A)**

```
class Animal {
    void sound() {
       System.out.println("Some sound");
    }
}

class Dog extends Animal {
    void sound() {
       System.out.println("Bark");
    }
}

Animal animal = new Dog();
animal.sound();
```

- **B)**

```
class Vehicle {
     void start() {
        System.out.println("Vehicle starting");
     }
}

class Car extends Vehicle {
       void start() {
        System.out.println("Car starting");
    }
}

Car car = new Car();
car.start();
```

- **C)**

```
class Shape {
    void draw() {
        System.out.println("Drawing");
    }
}

Shape shape = new Shape();
shape.draw();
```

- **D)**

```
interface Drawable {
    void draw();
}

class Circle implements Drawable {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

Circle circle = new Circle();
circle.draw();
```


Answer: A
Explanation: Polymorphism allows the method call to resolve at runtime. In option A, the sound() method demonstrates polymorphism, as the Animal reference holds a Dog object and calls Dog's sound() method.

Q#10. What is the main goal of dependency injection in OOP?

- **A.** To reduce code duplication
- **B.** To allow for dynamic method binding
- **C.** To reduce coupling between classes

- **D.** To allow classes to inherit each other's attributes

Answer: C
Explanation: Dependency injection is used to reduce coupling between classes by injecting dependencies rather than creating them within the class, promoting flexibility and testability.

Q#11. Which of the following terms best describes the relationship between a superclass and a subclass?

- **A.** "Depends-on" relationship
- **B.** "Has-a" relationship
- **C.** "Uses-a" relationship
- **D.** "Is-a" relationship

Answer: D
Explanation: Inheritance describes an "is-a" relationship, where a subclass is a specialized form of its superclass, sharing attributes and behaviors.

Q#12. Which of the following statements are true about abstraction in OOP?

1. Abstraction focuses on hiding the implementation details.
2. Abstract classes cannot contain any implemented methods.
3. Interfaces are a way to achieve abstraction.
4. Only abstract classes can be instantiated.

- **A.** 1 & 3
- **B.** 2 & 4
- **C.** 1, 2 & 3
- **D.** 1, 3 & 4

Answer: A
Explanation: Abstraction focuses on hiding unnecessary details (1) and can be achieved through interfaces or abstract classes (3). Abstract classes can contain implemented methods, and they cannot be instantiated.

Q#13. If a class Animal has a method speak() that is overridden in the subclass Dog, which concept allows the speak() method to call the subclass's implementation when an Animal reference holds a Dog object?

- **A.** Static Binding
- **B.** Dynamic Binding
- **C.** Overloading
- **D.** Aggregation

Answer: B
Explanation: Dynamic binding (or late binding) allows the method to be resolved at runtime, so the

Dog class's implementation of speak() will be called.

Q#14. Identify which code snippets correctly show method overloading.

- **A)**

```
public void printInfo(int id) { }
public void printInfo(String name) { }
```

- **B)**

```
public int calculateArea(int side) { return side * side; }
public int calculateArea(int length, int width) { return length * width; }
```

- **C)**

```
public void display() { }
public void display() { System.out.println("Hello"); }
```

- **D)**

```
public void setDetails(int age) { }
public void setDetails(int age, String name) { }
```


Answer: A, B, D
Explanation: Method overloading occurs when multiple methods in the same class have the same name but different parameters. Options A, B, and D meet this condition, while C does not due to identical parameter lists.

Q#15. In which of the following relationships does one class contain another class, and the contained class's existence is dependent on the container?

- **A.** Association
- **B.** Aggregation
- **C.** Inheritance
- **D.** Composition

Answer: D
Explanation: Composition implies a strong relationship, where the lifecycle of the contained object depends on the lifecycle of the containing object.

Q#16. Which relationship type is present if a Teacher class and Course class are associated but independent of each other?

- **A.** Composition
- **B.** Aggregation
- **C.** Association
- **D.** Inheritance

Answer: C
Explanation: Association represents a relationship between classes that are connected but independent in lifecycle.

Q#17. Which of the following is an example of static binding?

- **A.** Overloading methods
- **B.** Overriding methods
- **C.** Using interfaces
- **D.** Creating instances of abstract classes

Answer: A
Explanation: Static binding (compile-time binding) occurs with overloaded methods where the compiler determines the method to be called based on the method signature.

Q#18. Which code correctly defines a constructor for the Student class?

- **A)**

```
public void Student() { }
```

- **B)**

```
public Student() { }
```

- **C)**

```
private Student() { }
```

- **D)**

```
void Student() { }
```

Answer: B, C
Explanation: Constructors must have the same name as the class and no return type. Options B and C are valid constructors for Student.

Q#19. In an e-commerce system, a Payment interface is defined with a processPayment() method. Different payment types (like CreditCardPayment and PayPalPayment) implement this interface. Which concept does this demonstrate?

- **A.** Encapsulation
- **B.** Abstraction
- **C.** Inheritance
- **D.** Dependency

Answer: B
Explanation: The Payment interface abstracts the processPayment() behavior, allowing different implementations without specifying the actual details.

Q#20. What is the main purpose of a destructor in OOP?

- **A.** To initialize objects
- **B.** To provide additional functionalities to methods
- **C.** To clean up resources before an object is destroyed
- **D.** To encapsulate data

Answer: C
Explanation: Destructors are used to release resources (like memory) when an object's lifecycle ends, especially in languages without garbage collection.

Q#21. Which of the following terms best describes the relationship when one class depends on another to perform its functions?

- **A.** Aggregation
- **B.** Dependency
- **C.** Composition
- **D.** Polymorphism

Answer: B
Explanation: Dependency describes a relationship where one class requires another to perform its functions, often implemented through parameters or method calls.

Q#22. What is true about interfaces in OOP?

- **A.** Interfaces can't contain static fields
- **B.** Interfaces are used for encapsulation
- **C.** Interfaces can be instantiated directly
- **D.** Methods in interfaces have no implementation

Answer: D
Explanation: Interfaces define method signatures without implementation, providing a way to enforce certain behaviors without detailing how they are executed. Fields in interfaces are by default public static final.

Q#23. Which of the following situations exemplifies dynamic binding?

- **A.** A superclass reference variable pointing to a subclass object
- **B.** Invoking overloaded methods at compile time
- **C.** A constructor setting initial values for fields
- **D.** Creating an interface implementation

Answer: A
Explanation: Dynamic binding allows a superclass reference to invoke overridden methods in the subclass at runtime, that enables polymorphic behavior.

Q#24. Which code snippet demonstrates polymorphism?

- **A)**

```
Animal animal = new Dog();
animal.speak();
```

- **B)**

```
Dog dog = new Dog();
dog.speak();
```

- **C)**

```
Animal animal = new Animal();
animal.speak();
```

- **D)**

```
Dog dog = (Dog) new Animal();
dog.speak();
```


Answer: A
Explanation: Polymorphism is demonstrated by using a superclass reference (Animal) to refer to a subclass object (Dog) and invoking overridden methods.

Q#25. Which of the following options correctly describes encapsulation in OOP?

- **A.** Allowing one interface to be used for a general class of actions
- **B.** Inheriting properties and methods from a parent class
- **C.** Binding data and methods that manipulate the data within a single unit
- **D.** Enforcing unique behaviors in each subclass

Answer: C
Explanation: Encapsulation is the process of bundling data and methods that operate on that data within a single unit, typically through a class. It helps protect data from unauthorized access.

Q#26. In the code snippet below, what is myDog?

```
class Dog {
    String breed;
}

Dog myDog = new Dog();
```

- **A.** A class that holds details of various dogs
- **B.** An instance (object) of the Dog class
- **C.** A method to create dog breeds
- **D.** A variable that defines a dog's breed

Answer: B
Explanation: myDog is an **instance** (object) of the Dog class, created using the new keyword to instantiate the Dog class and allocate memory for myDog.

Q#27. A class Employee has properties and methods common to all employees, while subclasses like Manager and Developer inherit these common properties but add specific behaviors. What concept does this best illustrate?

- **A.** Encapsulation
- **B.** Inheritance
- **C.** Polymorphism
- **D.** Aggregation

Answer: B
Explanation: Inheritance allows subclasses to inherit common attributes and behaviors from a superclass, making it easier to manage shared code and add specific functionality in each subclass.

Q#28. A system includes an interface Shape with a method draw(). Classes Circle, Square, and Triangle implement Shape, each providing its own implementation of draw(). What concept allows the program to call draw() on any Shape object, regardless of its specific type?

- **A.** Inheritance
- **B.** Polymorphism
- **C.** Encapsulation
- **D.** Composition

Answer: B
Explanation: Polymorphism allows a single interface to be used with different implementations, enabling the program to invoke draw() without needing to know the specific type of Shape.

Q#29. Which option represents the "depends on" relationship in OOP?

- **A.** Aggregation
- **B.** Inheritance
- **C.** Dependency
- **D.** Composition

Answer: C
Explanation: Dependency represents a weak relationship where one class depends on another to function, but this dependency does not imply ownership or control.

Q#30. Which of the following relationships describes a "has-a" relationship where the lifecycle of the contained objects is independent?

- **A.** Aggregation
- **B.** Association
- **C.** Composition
- **D.** Dependency

Answer: A
Explanation: Aggregation represents a "has-a" relationship, where the containing object does not own the lifecycle of the contained object; they can exist independently.

Q#31. Which of the following is true about an interface in Java?

- **A.** An interface can contain static and default methods.
- **B.** An interface can't be instantiated directly.
- **C.** An interface can contain only method signatures and constants.
- **D.** An interface enforces private access on its methods.

Answer: B
Explanation: An interface in Java till JDK 7 contains only method signatures (abstract methods) and

constants (public, static, final fields), making it a contract for classes that implement it. JDK 8 onward as a new features introduced in Java 8, an interface can contain static & default methods as well. Default & static methods have implementation (method body).

Q#32. Which statement best defines message passing in OOP?

- **A.** Sending requests to other classes to execute specific functions
- **B.** Directly accessing another class's private variables
- **C.** Cloning objects across classes
- **D.** Binding methods statically during compile time

Answer: A
Explanation: Message passing refers to the process where objects communicate with each other by invoking methods, enabling interaction within an object-oriented system.

Q#33. Which of the following constructors is correctly written for the class Book?

- **A)**

```
public void Book() { }
```

- **B)**

```
public Book() { }
```

- **C)**

```
Book public() { }
```

- **D)**

```
public int Book() { }
```

Answer: B
Explanation: Constructors must have the same name as the class and cannot have a return type, making option B the only valid constructor.

Q#34. Given the following code, which statement best describes Person?

```
public class Person {
    String name;
    int age;
}
```

- **A.** Person is an instance of a class
- **B.** Person is a class that serves as a template for creating objects
- **C.** Person is an interface with attributes name and age
- **D.** Person is an enumeration of attributes

Answer: B
Explanation: Person is a class that defines attributes (name and age) and serves as a template for creating Person objects.

Q#35. A developer creates a class BankAccount where the balance field is private and can only be accessed and modified using methods deposit() and withdraw(). Which concept is being applied here?

- **A.** Inheritance
- **B.** Polymorphism
- **C.** Aggregation
- **D.** Encapsulation

Answer: D
Explanation: Encapsulation is used to restrict direct access to certain properties, enforcing controlled access through methods.

Q#36. Which of the following is an example of static binding?

- **A.** Method overriding
- **B.** Method overloading
- **C.** Dynamic polymorphism
- **D.** Abstract method implementation

Answer: B
Explanation: Static binding occurs at compile-time, and method overloading is an example since the compiler resolves the method call based on the method signature.

Q#37. A software module defines an abstract class Vehicle with an abstract method startEngine(). Concrete subclasses like Car and Bike provide specific implementations of this method. What concept does this demonstrate?

- **A.** Encapsulation
- **B.** Inheritance
- **C.** Abstraction
- **D.** Aggregation

Answer: C
Explanation: Abstraction involves defining a method in a generic way, allowing specific classes to implement their versions. This hides unnecessary details and allows focus on essential features.

Q#38. A Library class contains multiple Book objects. If a Library is deleted, all Book objects are also deleted. What relationship does this illustrate?

- **A.** Aggregation
- **B.** Composition
- **C.** Inheritance
- **D.** Dependency

Answer: B
Explanation: Composition indicates a strong relationship where the lifecycle of the contained objects (Book) is tied to the lifecycle of the container (Library).

Q#39. Which statement accurately defines polymorphism?

- **A.** Creating a class from another class
- **B.** Using a class solely as a data holder
- **C.** Enforcing access restrictions on class members
- **D.** Allowing multiple forms of a method or an object

Answer: D
Explanation: Polymorphism allows objects to be treated as instances of their parent class, enabling methods to have multiple forms based on context.

Q#40. A software developer designs a Database interface with methods like connect(), disconnect(), and executeQuery(). Classes like MySQLDatabase and MongoDatabase implement these methods. What principle is demonstrated here?

- **A.** Encapsulation

- **B.** Interface-based design
- **C.** Inheritance
- **D.** Aggregation

Answer: B
Explanation: Interface-based design enables classes to define specific behaviors according to a common contract (interface), promoting flexibility and standardization in application design.

Q#41. Which of the following statements best defines abstraction in OOP?

- **A.** Combining data and methods within a single unit
- **B.** Ensuring different classes have different implementations of a method
- **C.** Allowing multiple classes to inherit from a single superclass
- **D.** Hiding the complexity of certain operations and showing only essential features

Answer: D
Explanation: Abstraction in OOP is about hiding the complex implementation details and exposing only the essential features, allowing the user to focus on what an object does rather than how it does it.

Q#42. Which of the following best describes method overloading?

- **A.** Methods in different classes with the same name and parameters
- **B.** A method in a subclass with the same name as in the superclass
- **C.** Multiple methods in the same class with the same name but different parameters
- **D.** A method in an interface that is redefined in its implementing classes

Answer: C
Explanation: Method overloading occurs within the same class, where methods share the same name but differ in their parameters (number or type).

Q#43. A developer needs to initialize a class Car with a specific make and model as soon as an instance of Car is created. What should the developer implement in Car?

- **A.** Setter method
- **B.** Constructor
- **C.** Static method
- **D.** Destructor

Answer: B
Explanation: A constructor is used to initialize an object upon creation, allowing specific attributes to be set when the instance is created.

Q#44. In programming languages like C++, what purpose does a destructor serve?

- **A.** It creates an object
- **B.** It resets all object properties to their default values
- **C.** It deletes an object's memory when it's no longer needed
- **D.** It clones an object

Answer: C
Explanation: A destructor is used in languages like C++ to clean up resources and free memory when an object is no longer needed, ensuring efficient memory management.

Q#45. Which of the following best describes dynamic binding?

- **A.** Method resolution occurs at runtime
- **B.** Method resolution occurs at compile time
- **C.** Binding a static method to a class at compile time
- **D.** Linking a variable to a memory location during program startup

Answer: A
Explanation: Dynamic binding is the process of resolving method calls at runtime, which is common in polymorphic behavior where a specific implementation is selected based on the object type.

Q#46. In an online shopping system, a Customer can place multiple Order objects, but Order objects do not own the Customer object. What type of relationship does this describe?

- **A.** Composition
- **B.** Aggregation
- **C.** Association
- **D.** Dependency

Answer: C
Explanation: Association describes a relationship where objects interact without ownership or lifecycle control. Here, Customer and Order interact but do not depend on each other for lifecycle management.

Q#47. Consider the following code:

```
class Animal {
    public void sound() {
       System.out.println("Animal sound");
    }
}

class Dog extends Animal {
```

```
    @Override
    public void sound() {
        System.out.println("Woof");
    }
}

class Cat extends Animal {

    @Override
    public void sound() {
        System.out.println("Meow");
    }
}
```

What concept allows different classes (Dog and Cat) to implement their own version of the sound() method?

- **A.** Method Overloading
- **B.** Method Overriding
- **C.** Inheritance
- **D.** Encapsulation

Answer: B
Explanation: Method overriding allows subclasses to provide a specific implementation of a method that is defined in their superclass, enabling polymorphic behavior when sound() is called on an Animal reference.

Q#48. Which of the following scenarios best describes a dependency relationship?

- **A.** A Teacher has multiple Student objects, and the Student objects can exist independently
- **B.** A User class that takes a LoginService object to authenticate the user
- **C.** A Library object that contains multiple Book objects that do not depend on the library
- **D.** An Order class that contains an Item object and manages its lifecycle

Answer: B
Explanation: Dependency refers to a relationship where one class relies on another to function. In this case, the User class depends on LoginService for authentication.

Q#49. Which of the following examples is a case of static binding?

- **A.** A draw() method is called on different shapes at runtime
- **B.** A variable is linked to a memory address dynamically

- **C.** Method overloading within a single class
- **D.** A constructor is called during object creation

Answer: C
Explanation: Static binding occurs at compile time, as in the case of method overloading, where the compiler determines the method to call based on the parameters.

Q#50. Which relationship is described as a "part-of" relationship with lifecycle dependency?

- **A.** Dependency
- **B.** Association
- **C.** Composition
- **D.** Aggregation

Answer: C
Explanation: Composition represents a strong "part-of" relationship where the contained object's lifecycle is linked to the container object's lifecycle. If the container is destroyed, so its parts are also destroyed.

Q#51. Which OOP concept ensures that data is only accessible through specific methods in a class?

- **A.** Inheritance
- **B.** Polymorphism
- **C.** Encapsulation
- **D.** Abstraction

Answer: C
Explanation: Encapsulation restricts direct access to data within a class, allowing access only through specific methods. This ensures better control over data manipulation and integrity.

Q#52. Which component in the code below is responsible for defining the structure and behaviors of Book?

```
class Book {
    String title;
    String author;

    void read() {
        System.out.println("Reading " + title);
    }
}
```

- **A.** Book object
- **B.** Book class
- **C.** title variable
- **D.** read() method

Answer: B
Explanation: The Book class defines the blueprint of the Book object, including its properties (title, author) and behaviors (read() method).

Q#53. In the following code, what is myCar?

```
Car myCar = new Car();
```

- **A.** Method
- **B.** Class
- **C.** Object
- **D.** Constructor

Answer: C
Explanation: myCar is an instance (object) of the Car class, created using the new keyword which allocates memory and calls the constructor.

Q#54. Consider the following code. Which of these statements are correct about Laptop laptop1 = new Laptop("Dell");?

```
public class Laptop {
    private String brand;

    public Laptop(String brand) {
        this.brand = brand;
    }
}

Laptop laptop1 = new Laptop("Dell");
```

- **A.** laptop1 is an instance of the Laptop class with its brand initialized to "Dell".
- **B.** The new keyword creates a new class template Laptop.

- **C.** The constructor Laptop(String brand) initializes the brand attribute of laptop1.
- **D.** Laptop must extend another class for laptop1 to be initialized.

Answer: A, C
Explanation: A and **C** are correct as laptop1 is an instance with brand initialized through the constructor. **B** is incorrect as the new keyword creates an object, not a class. **D** is incorrect as there's no requirement for inheritance here.

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/ccee2025notes

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project