1. What is java?
2. What is JDK?
3. Draw the JDK diagram?
4. How many data types we have ?
5. Ascending order of numeric data types?
6. What is type casting?
7. What is JVM architecture?
8. How many class loaders we have ?
9. How many . dot extensions will create after executing the one code?
10. Can I take multi classes in single java file?
11. How many token we have ?
12. What are access modifiers we have?
13. How many access modifiers tokens we have?
14. Explain class structure?(state and behavior of PEN class).
15. Description of main method.
16. What is meant by "Write Once, Run Anywhere" in the context of Java?
17. Explain how Java achieves platform independence.
18. What role does the Java Virtual Machine (JVM) play in making Java platform-independent?
19. How do byte code and the Java compiler contribute to platform independence?
20. Explain the process of executing a Java program on different platforms.
21. What is the difference between platform independence and platform-specific features in Java?
22. What is a literal in Java?
23. How are integer literals represented in Java?
24. Explain the difference between a decimal literal and an octal literal in Java.
25. What is a floating-point literal, and how is it written in Java?
26. What is an instance variable in Java, and how does it differ from a local variable?
27. Explain the concept of a static variable. How is it different from an instance variable?
28. What is a parameter variable in the context of methods in Java? How are parameter variables used?
29. Define local variable in Java. How is the scope of a local variable determined, and what are its limitations?
30. Discuss the scope of instance variables, static variables, parameter variables, and local variables. How does the scope of each type of variable affect its visibility in different parts of a Java program?
31. How can you initialize an instance variable in Java with a user-defined value during object creation?
32. Explain how constructors can be utilized to set user-defined values for instance variables in a Java class.
33. What is the role of setter methods, and how do they facilitate the assignment of user-defined values to instance variables?
34. How can user input be processed in Java, and how might this input be used to initialize or modify instance variables within an object?
35. What is the purpose of the this keyword in Java, and how is it used?
36. In the context of an instance method, explain how and why the this keyword is employed.
37. How is the this keyword used within a constructor? Provide an example to illustrate its usage.
38. Discuss the concept of method chaining in Java and explain how the this keyword can be involved in this process.
39. Explain the concept of data hiding in Java and its significance in object-oriented programming.
40. Define abstraction in the context of Java programming. How does it contribute to the design of software?
41. What is an abstract class in Java? How does it differ from a regular (concrete) class?
42. Explain the concept of abstract methods. How are they declared and implemented in Java?
43. Compare and contrast abstract classes and interfaces in Java. When would you prefer using one over the other?
44. Achieving Abstraction:
45. How can you achieve abstraction in Java through the use of abstract classes, interfaces, and abstract methods? Provide examples to illustrate your answer.
46. What is encapsulation in Java, and why is it considered a fundamental principle of object-oriented programming?

47. Explain the role of access modifiers (e.g., private, public, protected) in achieving encapsulation in Java.
48. How do getter and setter methods contribute to encapsulation? Provide an example to illustrate their usage.
49. Enumerate the benefits of encapsulation in Java, emphasizing the advantages it brings to code maintainability and security.
50. Discuss how encapsulation is applied in real-world scenarios, providing examples from Java libraries or frameworks that demonstrate the use of encapsulation principles.
51. What is a constructor in Java? How does it differ from a regular method?
52. what are the types of Constructor?
53. Explain the concept of a default constructor. When is it automatically provided by the compiler?
54. How do you define a parameterized constructor in Java? Provide an example.
55. What is constructor overloading? Provide a scenario where constructor overloading would be useful.
56. Discuss the role of constructors in initializing objects. How are constructors invoked during object creation?
57. Compare and contrast the use of ==, equals(), and hashCode() when comparing objects in Java. Under what circumstances would you override these methods in a class?
58. What is the purpose of the equals(Object o) method? How can it be overridden to provide a meaningful comparison between objects of a class?
59. Discuss the role of the toString() method in Java. How does it contribute to the readability and debugging of code?
60. Explain the significance of the hashCode() method. How is it used in Java, and what considerations should be taken into account when implementing this method?
61. In what situations would you commonly use the toString() method to represent an object as a string?
62. Describe the difference between reference equality (==) and the equals(Object o) method. When would you override the equals() method to achieve a meaningful comparison of object contents?
63. What is the IS-A relation in Java, particularly in the context of inheritance?
64. How does inheritance promote code reuse in Java?
65. Explain the concept of inheritance and its role in object-oriented programming.
66. How is the IS-A relationship established between classes in Java?
67. List and briefly explain the different types of inheritance supported in Java.
68. What is the difference between single and multiple inheritance?
69. How are this() and super() used in Java constructors?
70. Explain the significance of calling this() and super() in a constructor.
71. Discuss the reasons why Java does not support multiple inheritance.
72. Describe the access modifiers in Java and their significance.
73. How do access modifiers contribute to encapsulation in Java?
74. What is the HAS-A relation in Java, specifically in the context of association between classes?
75. Provide an example of a HAS-A relationship in a Java program.
76. Define superclass and subclass in the context of inheritance.
77. How is the superclass related to the subclass in terms of syntax structure?
78. Differentiate between method overriding and method overloading.
79. Provide examples to illustrate the concepts of method overriding and overloading.
80. Explain how polymorphism is achieved through inheritance in Java.
81. Discuss the advantages of polymorphism in object-oriented programming.
82. what is defference between compiletime polymorphisam and static polymorphisam?
83. what is defference between runtime polymorphisam and dynamic polymorphisam?
84. Compare and contrast abstract classes and interfaces in Java.
85. When would you choose to use an abstract class over an interface, and vice versa?
86. What is covariant?
87. Explain the concept of method hiding in Java inheritance.
88. When does method hiding occur, and how is it different from method overriding?
89. Describe the role of the protected access modifier in inheritance.
90. What is Autoboxing in Java?
91. Explain the concept of Unboxing in Java.

92. How does Autoboxing simplify code in Java?
93. What are the primitive data types that are involved in Autoboxing?
94. Provide an example of Autoboxing in Java.
95. Give an example of Unboxing in Java.
96. What is the role of the wrapper classes in Autoboxing and Unboxing?
97. Can Autoboxing lead to performance overhead? Explain.
98. How does Autoboxing help when working with collections in Java?
99. Discuss situations where Autoboxing and Unboxing might lead to NullPointerException.
100. What is the purpose of the final keyword in Java?
101. How does declaring a variable as final affect its value?
102. Explain the significance of a final method in Java.
103. Can a final variable be reassigned after its initial value is set?
104. Describe the role of the final keyword in the context of classes.
105. In Java, how does the final keyword relate to constants?
106. Can a method be both static and final in Java? Explain.
107. What is the difference between a final variable and a final method?
108. How does the use of the final keyword contribute to code optimization?
109. Explain the behavior of a final class in terms of inheritance.
110. What is a nested inner class in Java, and how does it differ from a regular (non-nested) class? Provide an example.
111. Explain the concept of a method-local inner class. In what scenarios might you use this type of inner class?
112. Define a static nested inner class in Java. How is it different from a non-static nested inner class?
113. What is an anonymous inner class, and how is it used in Java? Provide a scenario where anonymous inner classes are particularly useful.
114. Discuss the relationships between All inner classes. How do they interact with their enclosing classes and each other?
115. How do you declare an enum in Java, and what is the significance of using enums over other data types?
116. Explain the concept of enum constants. How are they defined within an enum, and what type of values can they represent?
117. Discuss the role of methods in enums. How can you add custom methods to an enum, and what purpose might they serve?
118. Name and briefly explain the main components of the Java Virtual Machine (JVM) architecture.
119. What is the role of the class loader in the JVM architecture? How does it contribute to the loading of Java classes?
120. Describe the purpose of the execution engine in the JVM. How does it execute Java bytecode?
121. Identify and explain the different memory areas in the JVM architecture. How is memory management handled during the execution of a Java program?

Programs :

1.Find output of this program?

```
public class InfiniteExample {
static int methodName(  ) {
System.out.println("a value is :"+(a=a));
a= ++a+(a=++a+--b+c*(--c+a++)); //2+(3+1+3(2+3))  //2+(3+1+15)// 21
return a;
}
static int sub3values(int a, int b) {
return a=--a+(b=--b-(--a+b--)+--a-++b*(++b+--b));
}
public static void main(String[] args) {
System.out.println(methodName(1, 2, 3));
System.out.println(sub3values(10, 5));
}
}
```

2. Write the code for reversing the string?
3. Write the code for count the char's from given numbers?
4. Write the code for count of 1's and count of 0's from string?
5. Write the code for fibonacc?
6. Write the code for some patterns?
7. Find Output

```
        public class InfiniteExample {
        char ch='A';
        public static void main(String[] args) {
        System.out.println(c+1);
        System.out.println(c++);
        System.out.println(++c);
}
```

8. write code for == and equals() and explain?
9. Write the code for constructor overloading ?
10. Find the which syntax's are correct

```
public static void methodName( ) {-----}
static public void methodName( ) {-----}
static void public methodName( ) {-----}
void static public methodName( ) {-----}
```