Comprehensive OOPs Interview Questions List for Tech Interviews

Based on extensive research from multiple sources, here's a complete compilation of **186 Object-Oriented Programming interview questions** specifically curated for **tech**, **software development**, **machine learning**, and **deep learning** engineering positions.

Basic OOP Concepts (16 Questions)

Fundamental Understanding

1. What is Object-Oriented Programming (OOP)?[1] [2] [3]
2. What are the four main principles of OOP?[4] [2] [5]
3. What is a class?[2] [3] [6]
4. What is an object?[3] [6] [2]
5. What is the difference between a class and an object?[7] [4] [2]
6. What is the need for OOPs?[5] [8] [2]
7. What are the advantages of OOPs?[9] [2] [5]
8. What are the disadvantages of OOPs?[10] [8] [2]
9. What is a constructor?[6] [2] [5]
10. What is a destructor?[2] [9] [5]
11. What are the different types of constructors?[5] [10] [2]
12. What is the difference between constructor and method?[11] [10] [2]
13. What is the purpose of 'this' keyword?[12] [10] [5]
14. What is a static method?[12] [6] [5]
15. What is an instance method?[6] [11] [12]
16. What is the difference between static and instance methods?[7] [11] [5]

Encapsulation (10 Questions)

Data Hiding & Access Control

17. What is encapsulation?[1] [2] [5]
18. What are access modifiers/specifiers?[8] [10] [2]
19. Explain public, private, and protected access modifiers[4] [8] [2]
20. What is information hiding in OOP?[13] [8] [12]
21. What are getter and setter methods?[2] [5] [12]

22. How does encapsulation improve security?[8] [5] [12]
23. What is data abstraction?[9] [3] [5]
24. How is encapsulation different from data abstraction?[14] [11] [5]
25. Can you give a real-life example of encapsulation?[5] [12] [8]
26. How do you implement encapsulation in Python?[13] [12] [6]

Inheritance (18 Questions)

Code Reusability & Hierarchies

27. What is inheritance?[1] [12] [2]
28. What are the types of inheritance?[9] [6] [2]
29. What is single inheritance?[8] [2] [9]
30. What is multiple inheritance?[12] [6] [5]
31. What is multilevel inheritance?[2] [9] [8]
32. What is hierarchical inheritance?[9] [8] [2]
33. What is hybrid inheritance?[8] [2] [9]
34. What is a superclass/base class?[11] [2] [9]
35. What is a subclass/derived class?[11] [12] [9]
36. What is the difference between multiple and multilevel inheritance?[2] [9] [8]
37. What is the use of inheritance?[12] [8] [2]
38. How can you prevent inheritance in a class?[10] [8] [2]
39. What is the super keyword?[6] [5] [12]
40. Can you call base class method without creating an instance?[5] [11] [2]
41. What are the limitations of inheritance?[10] [9] [8]
42. What is composition vs inheritance?[15] [11] [8]
43. When to use inheritance vs composition?[15] [11] [8]
44. What is the diamond problem in multiple inheritance?[13] [6] [12]

Polymorphism (17 Questions)

Method Behavior Variations

45. What is polymorphism?[1] [5] [2]
46. What are the types of polymorphism?[9] [5] [2]
47. What is compile-time polymorphism?[5] [8] [2]
48. What is runtime polymorphism?[8] [2] [5]
49. What is method overloading?[4] [2] [5]
50. What is method overriding?[4] [2] [5]
51. What is the difference between overloading and overriding?[2] [5] [8]
52. What is operator overloading?[14] [9] [5]
53. What is dynamic binding?[5] [8] [2]
54. What is static binding?[8] [2] [5]
55. What is a virtual function?[9] [2] [5]
56. What is a pure virtual function?[9] [5] [8]
57. How is runtime polymorphism achieved?[2] [5] [8]
58. What is early binding vs late binding?[11] [8] [2]
59. Can you override static methods?[11] [5] [8]

60. Can you override private methods?[10] [11] [8]
61. What is method hiding vs method overriding?[15] [11] [8]

Abstraction (16 Questions)

Hiding Implementation Complexity

62. What is abstraction?[1] [5] [2]
63. What is an abstract class?[10] [5] [2]
64. What is an abstract method?[10] [5] [2]
65. What are the characteristics of abstract class?[10] [8] [2]
66. Can you instantiate an abstract class?[5] [2] [10]
67. What is an interface?[3] [2] [5]
68. What is the difference between abstract class and interface?[8] [2] [5]
69. Can an abstract class have constructor?[11] [10] [8]
70. Can an abstract class have static methods?[2] [10] [8]
71. Can abstract methods be private?[11] [10] [8]
72. When to use abstract class vs interface?[5] [8] [11]
73. Can interface extend multiple interfaces?[8] [11] [2]
74. Can class implement multiple interfaces?[11] [2] [8]
75. What is the difference between extends and implements?[10] [2] [8]
76. How is abstraction accomplished?[2] [8] [11]
77. What is the purpose of abstract classes?[5] [8] [11]

Advanced OOP Concepts (19 Questions)

Design Patterns & Best Practices

78. What is a friend function?[16] [9] [2]
79. What is exception handling in OOP?[10] [2] [5]
80. What is a try-catch block?[2] [5] [10]
81. What is a finally block?[9] [5] [10]
82. What is the finalize method?[5] [10] [2]
83. What is garbage collection?[8] [10] [5]
84. What is casting in OOP?[11] [8] [2]
85. What is upcasting and downcasting?[15] [8] [11]
86. What is a singleton pattern?[15] [9] [11]
87. What is a factory pattern?[15] [8] [11]
88. What is coupling in OOP?[17] [8] [11]
89. What is cohesion in OOP?[15] [8] [11]
90. What is SOLID principles?[3] [8] [11]
91. What is dependency injection?[15] [8] [11]
92. What is the Liskov substitution principle?[15] [8] [11]
93. What is the open/closed principle?[8] [11] [15]
94. What is the single responsibility principle?[11] [15] [8]
95. What is interface segregation principle?[15] [8] [11]
96. What is dependency inversion principle?[8] [11] [15]

Python-Specific OOP Questions (15 Questions)

Python Implementation Details

97. What is `__init__` method?[6] [13] [12]
98. What is `__del__` method?[18] [12] [6]
99. What is the difference between `__str__` and `__repr__`?[14] [13] [6]
100. What is multiple inheritance in Python?[13] [12] [6]
101. What is the Method Resolution Order (MRO)?[14] [6] [13]
102. What are decorators in Python OOP?[12] [6] [13]
103. What is the `@property` decorator?[13] [14] [12]
104. What is the `@staticmethod` decorator?[6] [12] [13]
105. What is the `@classmethod` decorator?[12] [6] [13]
106. What is duck typing in Python?[14] [6] [13]
107. What are magic methods/dunder methods?[6] [13] [14]
108. What is the `super()` function in Python?[13] [12] [6]
109. What is the difference between old-style and new-style classes?[14] [6] [13]
110. What is metaclass in Python?[6] [13] [14]
111. What is the `__new__` method?[13] [14] [6]

Practical Scenario Questions (15 Questions)

Real-World Design Problems

112. Design a banking system using OOP principles[11] [15] [8]
113. How would you model a university system?[15] [8] [11]
114. Design a library management system[8] [11] [15]
115. How would you implement a shape hierarchy?[11] [15] [8]
116. Design a vehicle management system[1] [8] [11]
117. How would you model an e-commerce system?[15] [8] [11]
118. Design a game character system[8] [11] [15]
119. How would you implement a file system?[11] [15] [8]
120. Design a social media platform structure[15] [8] [11]
121. How would you model a hospital management system?[8] [11] [15]
122. Design a music streaming service[11] [15] [8]
123. How would you implement a chat application?[15] [8] [11]
124. Design a restaurant ordering system[8] [11] [15]
125. How would you model an employee management system?[11] [15] [8]
126. Design a booking system (hotel/flight/movie)[15] [8] [11]

Comparison Questions (15 Questions)

Conceptual Differences

127. Difference between procedural and object-oriented programming[12] [2] [8]
128. Difference between composition and inheritance[8] [11] [15]
129. Difference between aggregation and composition[11] [15] [8]
130. Difference between method overloading and method overriding[4] [2] [8]
131. Difference between abstract class and interface[2] [5] [8]
132. Difference between static and dynamic polymorphism[5] [2] [8]

133. Difference between early binding and late binding[2] [8] [11]
134. Difference between shallow copy and deep copy[4] [2] [8]
135. Difference between public, private, and protected[4] [2] [8]
136. Difference between class variable and instance variable[12] [13] [8]
137. Difference between constructor and destructor[2] [8] [11]
138. Difference between extends and implements[10] [2] [8]
139. Difference between final, finally, and finalize[10] [8] [11]
140. Difference between throw and throws[10] [8] [11]
141. Difference between method and function[15] [8] [11]

Machine Learning & AI Specific Questions (15 Questions)

OOP in Data Science Context

142. How do you use OOP principles in machine learning pipelines?[19] [20] [18]
143. Design a machine learning model class hierarchy[21] [18] [19]
144. How would you implement polymorphism in ML algorithms?[20] [18] [19]
145. Design a neural network using OOP concepts[18] [19] [20]
146. How do you apply encapsulation in data preprocessing?[19] [21] [18]
147. Create an abstract base class for different ML models[21] [18] [19]
148. How would you use inheritance for different types of neural networks?[20] [18] [19]
149. Design a data loader class for ML projects[22] [18] [21]
150. How do you implement the strategy pattern for different optimization algorithms?[19] [20] [21]
151. Design a feature engineering pipeline using OOP[22] [18] [21]
152. How would you create a model evaluation framework?[20] [21] [19]
153. Design a hyperparameter tuning system[22] [19] [20]
154. How do you implement different loss functions using polymorphism?[18] [19] [20]
155. Create a factory pattern for different ML algorithms[21] [18] [19]
156. How would you design a model versioning system?[18] [21] [22]

Advanced Technical Questions (31 Questions)

Deep Dive Topics

157. What is method resolution order in multiple inheritance?[14] [6] [13]
158. How do you implement custom iterators using OOP?[6] [13] [14]
159. What is the difference between composition and aggregation?[8] [11] [15]
160. How do you implement the observer pattern?[11] [15] [8]
161. What is the strategy pattern and when to use it?[15] [8] [11]
162. How do you implement thread-safe singleton?[8] [11] [15]
163. What is the template method pattern?[11] [15] [8]
164. How do you implement custom exceptions?[13] [12] [6]
165. What is the adapter pattern?[15] [8] [11]
166. How do you implement the decorator pattern?[8] [11] [15]
167. What is the facade pattern?[11] [15] [8]
168. How do you implement lazy initialization?[15] [8] [11]
169. What is the builder pattern?[8] [11] [15]
170. How do you implement the command pattern?[11] [15] [8]
171. What is the state pattern?[15] [8] [11]

172. How do you implement the visitor pattern?[8] [11] [15]
173. What is the proxy pattern?[11] [15] [8]
174. How do you implement the chain of responsibility pattern?[15] [8] [11]
175. What is the mediator pattern?[8] [11] [15]
176. How do you implement the memento pattern?[11] [15] [8]
177. What is the prototype pattern?[15] [8] [11]
178. How do you implement the interpreter pattern?[8] [11] [15]
179. What is the bridge pattern?[11] [15] [8]
180. How do you implement the composite pattern?[15] [8] [11]
181. What is the flyweight pattern?[8] [11] [15]
182. How do you handle circular dependencies in OOP?[11] [15] [8]
183. What is inversion of control?[15] [8] [11]
184. How do you implement aspect-oriented programming concepts?[8] [11] [15]
185. What is the difference between tight coupling and loose coupling?[11] [15] [8]
186. How do you design for testability in OOP?[15] [8] [11]

Interview Preparation Tips

For Tech Interviews:

- Focus on **practical implementation** rather than just theoretical knowledge[23] [24] [19]
- Practice **coding examples** for each concept in Python[12] [6] [13]
- Understand **design patterns** commonly used in software development[8] [11] [15]
- Be prepared to **write code** that demonstrates OOP principles[1] [2] [11]

For ML/AI Roles:

- Emphasize how OOP **improves code organization** in ML pipelines[19] [21] [18]
- Show understanding of **scalable architecture** for ML systems[22] [18] [19]
- Demonstrate knowledge of **design patterns** specific to data science[21] [18] [22]
- Practice **implementing ML algorithms** using OOP principles[20] [18] [19]

This comprehensive list covers all essential OOP concepts from basic to advanced levels, specifically tailored for technical interviews in software development, machine learning, and deep learning engineering positions.

⁂

1. https://www.interviewbit.com/oops-interview-questions/
2. https://prepinsta.com/interview-preparation/technical-interview-questions/oops/
3. https://www.geeksforgeeks.org/interview-prep/oops-interview-questions/
4. https://www.hirist.tech/blog/top-50-java-oops-interview-questions-and-answers/
5. https://www.simplilearn.com/tutorials/java-tutorial/oops-interview-questions
6. https://www.geeksforgeeks.org/python/python-oops-interview-question/
7. https://leetcode.com/discuss/interview-question/object-oriented-design/4440061/top-16-oops-interview-questions-with-answers

8. https://www.guvi.in/blog/object-oriented-programming-interview-questions-and-answers/
9. https://www.youtube.com/watch?v=5-9BhmP9Ufk
10. https://www.c-sharpcorner.com/article/oops-interview-questions-c-sharp/
11. https://www.finalroundai.com/blog/oops-interview-questions
12. https://in.indeed.com/career-advice/interviewing/python-oops-interview-questions
13. https://www.hirist.tech/blog/top-25-python-oops-interview-question-2024/
14. https://pynative.com/python-oop-interview-questions/
15. https://interviewkickstart.com/blogs/interview-questions/oops-interview-questions-for-experienced-programmers
16. https://www.interviewbit.com/cpp-interview-questions/
17. https://www.youtube.com/watch?v=R_uqDROtXSk
18. https://blog.gopenai.com/important-interview-questions-for-python-in-machine-learning-object-oriented-programming-concepts-9a71c1a76f0c
19. https://www.finalroundai.com/blog/deep-learning-engineer-interview-questions
20. https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-interview-questions
21. https://skillfloor.com/blog/machine-learning-engineer-interview-questions
22. https://www.guvi.in/blog/top-deep-learning-interview-questions-and-answers/
23. https://dl.acm.org/doi/10.1145/3626252.3630860
24. https://ieeexplore.ieee.org/document/10663022/