Inheritance

1. **What is the advantage of inheritance?**

Polymorphism, code re-usability and hence inheritance enhances profitability.

2. **What is a parent class also called as?**

Super class or base class.

3. **What is a child class also called in java?**

Derived class or sub class.

4. **Which relationship must be satisfied to achieve inheritance in java?**

"is-a" relationship.

5. **What is the meaning of extends keyword in java and when is it used?**

Meaning of extends keyword is "is-a". "extends" keyword would be used during inheritance.

6. **Can classes be circularly related in java?**

No. Cyclic dependency is not permitted in java.

7. **How do constructor execute in case of inheritance?**

Constructors do not participate in inheritance. However, constructors would get executed through constructor chaining using super() method. This rule is made in order to preserve encapsulation.

8. **What is the use of inheritance in java?**

Polymorphism, code re-usability and hence inheritance enhances profitability.

9. **Does java support multiple inheritance explain?**

No. Because diamond shaped problem would be occurred which leads to ambiguity.

(Eg. Refer class notes)

10. Does inheritance promote "is-a" relationship or "has-a" relationship?

Inheritance promotes "is-a" relationship.

11. Why does java not support multiple inheritance? Explain with example?

No. Because diamond shaped problem would be occurred which leads to ambiguity.

(Eg. Refer class notes)

12. Can constructors be inherited in java?

No, constructors cannot be inherited in java. However, the control would go from the derived class to the base class constructor using super() method and constructor of the base class gets executed. Therefore, even though constructors are not inherited, yet they are executed

13. Can we reduce the visibility of the inherited method?

No. Either the visibility of the inherited method must be the same as that of the parent or it should be having higher visibility.

14. Is ~~inheritance~~ aggregation loose bound or tight bound?

Loose bound.

15. Is composition loose bound or tight bound?

Tight bound.

16. Is it compulsory for the method to be inherited for it to be overridden?

Yes.

17. Can we override a private method? Why?

No. Private method would never get inherited and hence can never be overridden.

18. What is a co-variant return type?

Co-variant return types are such return types which have "is-a" relationship between themselves. Java permitted co-variant return types from JDK 1.5 onwards.

(Eg. Refer class notes)

19. What actually happens in inheritance?

The features present in the parent would be inherited by the child. The child would override such features of the parent which it is not interested in and would also adds a few specialized features.

20. Using extends keyword can a subclass inherit all the properties of super class?

No. However, except private members and constructors, all other properties of the super class can be inherited by the sub class.

21. How is instanceof operator used in inheritance?

"instanceof" operator is used to test whether the object is an instance of the specified class or not. It returns true if the object belongs to the specified class.

22. Can a subclass have any number of superclasses?

No. Because multiple inheritance is not supported in java.

23. Can a superclass have any number of subclasses?

Yes.

24. Do both properties and behaviors get inherited in java?

Yes. However, private members and constructers do not get inherited.

25. Which is the superclass of all classes in java?

Object class.

26. Which is the superclass of Object class in java?

Object class is the top most super class. There is no super class above it.

27. Does java support multi-level inheritance?

Yes.

28. In which package is the Object class present?

java.lang package.

29. What does the Object class contain?

Object class contains number of methods which are required for all the sub classes such as **toString(), hashCode(), getClass(), finalize() etc.**

30. In the class hierarchy which classes are more generalized in behavior?

In the hierarchical design, the classes present above in the hierarchy are generalized and as we flow down in the hierarchy they become more and more specific.

31. In the class hierarchy which classes are more specialized in behavior?

In the hierarchical design, the classes present above in the hierarchy are generalized and as we flow down in the hierarchy they become more and more specific.

32. Can we have a method in a subclass with a same name as in the superclass?

Yes. It is called as overriding.

33. Can we have a variable in a subclass with a same name as in the superclass?

Yes.

34. Can we declare new fields in the subclass which are not there in the superclass?

Yes. Such fields are called as specialized fields.

35. Why does not java permit constructor inheritance?

It is not permitted in order to preserve encapsulation.

36. What is the difference between hiding, overriding and shadowing?

Hiding refers to the process of programming the objects such that other objects cannot directly access the most important components of the current objects. Though direct access is prevented, controlled access is permitted through setters and getters. It is also referred to as encapsulation.

Overriding refers to the process of the subclass inheriting a method of the super class and modifying it to suit its specific needs.

Shadowing refers to the name clash that occurs when the local variable names and the instance variable names are the same. It can be overcome using "this" keyword.

37. Can we declare new methods in the subclass which are not there in the superclass?

Yes. Such methods are called specialized methods.

38. Can a subclass access private members of the superclass directly?

No.

39. What are the ways in which private members of a superclass can be accessed?

It can be accessed through public setters and getters.

40. What is not possible in java inheritance?

i) Private inheritance is not possible.
ii) Constructor inheritance is not possible.
iii) Multiple inheritance is not possible.
iv) Cyclic inheritance is not possible.

41. Comment on the visibility of private?

Visible only within the given class.

42. Comment on the visibility of protected?

Visible in the current package and also visible to the classes of other packages provided the classes in the other packages are sub classes of the current class.

43. Comment on the visibility of default?

Accessible in the current package.

44. Comment on the visibility of public?

Accessible throughout the project.

45. What is an alternative to inheritance?

Delegation model.

46. How is delegation better when compared to inheritance?

Though inheritance has many advantages, yet one of the limitations is that the subclass would be forced to inherit all the methods of the super class irrespective of whether the subclass requires them or not. If the subclass requires only few methods of the super class, then such an arrangement is not possible in inheritance. It can be achieved using delegation model.

However, since delegation model is not a hierarchical model, loose coupling cannot be achieved and hence polymorphism cannot be achieved.

47. Which relationship does inheritance implement?

"is-a" relationship.

48. Who promotes "is-a" relationship?

Inheritance.

49. Who promotes "has-a" relationship?

Aggregation and composition.

50. What is meant by implicit typecasting?

The process of placing a smaller magnitude of data into a larger data type automatically by the compiler without programmer intervention is called as implicit typecasting. It is also called as numeric promotion.

Eg. byte can automatically converted to short type.

51. What is meant by downcasting?

(Refer class notes)

52. What is meant by upcasting?

(Refer class notes)

53. How many levels above the current level would the super keyword enable to access in case of inheritance?

"super" keyword enables to access fields only one level above in the hierarchy.

54. Which constructor of the superclass gets called automatically from the subclass?

Default constructor.

55. Can we call the parameterized constructor of the subclass from the baseclass?

No.

56. Can a subclass somehow access the private members of the superclass?

Yes. Through public setters and getters.

57. What is constructor chaining?

It refers to the process of the subclass constructor calling its super class constructor by using super() method.

58. Can a parent reference point to the child object?

Yes.

59. What are the advantages of parent reference to the child object?

It results in loose coupling and hence enables to achieve polymorphism. Through this flexibility and code reduction also can be achieved.

60. What is meant by loose coupling?

Creating a parent reference to point to child object is called as loose coupling.

61. What is meant by tight coupling?

Creating a child reference to point to child object is called as tight coupling.

62. Is parent reference to child object loose coupling or tight coupling?

Loose coupling.

63. What is the limitation associated with parent reference to child object?

Using such a parent reference only the overridden methods present in the child class can be accessed directly. The specialized methods present in the child cannot be accessed directly.

64. How can the limitation of parent reference to child object be overcome?

By downcasting.

65. How can we prevent the inheritance of a class?

By making the class as final.

66. Why is cyclic dependency not permitted in java?

Because it does not exists in real life.

67. What are the advantages of method overriding?

Overriding enables the subclass to redesign the method to suit its specific requirements.

68. Why are private members not inherited?

To preserve the concept of encapsulation.

69. Why are constructors not inherited?

To preserve the concept of encapsulation.

70. When a derived class object is created, why does the constructor of base class get called?

Due to constructor chaining.

71. What is meant by diamond shape problem?

Multiple inheritance. Any project design must not entertain diamond shape problem since it would lead to ambiguity.

72. Does java support multi-level inheritance? Why?

Yes. Because it is existing in real life.

73. How is has-a relationship implemented in java?

Using composition, aggregation and delegation

74. What is composition?

It refers to tight bound has-a relationship.

75. What is aggregation?

It refers to loose bound has-a relationship.

76. Can we access the composite object while the enclosing object is accessible?

Yes.

77. Can we access the aggregate object while the enclosing object is accessible?

Yes.

78. Can we access the composite object while the enclosing object is not accessible?

No.

79. Can we access the aggregate object while the enclosing object is not accessible?

Yes.

80. Can you identify composite objects in this interview room?

Yes. Candidate has a heart, has a knowledge etc.

81. Can you identify aggregate objects in this interview room?

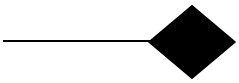
Yes. Candidate has a resume, has a book, has a pen etc.

82. Which classes in the UML diagram must be made final?

The leaf nodes.

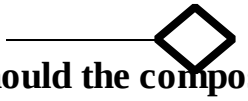
83. Which notation is used to represent composite objects in UML?

Diamond shape with coloring inside (shaded rhombus)



84. Which notation is used to represent aggregate objects in UML?

Diamond shape without coloring.



85. Why should the composite objects get destroyed when the enclosing object is destroyed?

Because that is the way real life objects behave.

86. Why should the aggregate objects not get destroyed when the enclosing object is destroyed?

Because that is the way real life objects behave.

87. Can we represent both "is-a" and "has-a" relationship in a single UML?

Yes.

88. Does the delegation model implement "is-a" relationship?

Not exactly. However, it can be used as an alternative to inheritance.

89. Does the delegation model implement "has-a" relationship?

Yes.

90. What happens if the class is made final?

It cannot be further inherited.

91. What happens if the method is made final?

It cannot be overridden.

92. What happens if the variable is made final?

It behaves like a constant.

93. Can you name an inbuilt class which is final?

String class

94. What is the casting from a general to a more specific type called as?

Downcasting.

95. What is the casting from a specific to a more general type called as?

Upcasting

96. What is the difference between primitive numeric type casting and casting between object references?

In primitive numeric typecasting we have two types of casting namely implicit typecasting and explicit typecasting. On object references, we do not have implicit and explicit typecasting, rather we have upcasting and downcasting concepts.