Data abstraction and Object Orientation

1. What are generally considered to be the three defining characteristics of object-oriented programming?

2. In what programming language of the 1960s does object orientation find its roots? Who invented that language? Summarize the evolution of the three defining characteristics since that time.

3. Name three important benefits of abstraction.

4. What are the more common names for subroutine member and data memebers?

5. What is a property in C#.

6. What is the purpose of the "private" part of an object interface? Why can't it be hidden completely?

7. What is the purpose of the `::` operator in C++?

8. Explain why in-line subroutines are particularly important in object-oriented languages.

9. What are constructors and destructors?

10. Give two other terms, each, for base class and derived class.

11. Explain why generics may be useful in an object-oriented language, despite the extensive polymorphism already provided by inheritance.

12. What is meant by an opaque export from a module?

13. What are private types in Ada?

14. Explain the significance of the `this` parameter in object-oriented languages.

15. How do Java and C# make do without explicit class headers?

16. Explain the distinctions among `private`, `protected`, and `public` class members in C++.

17. Explain the distinctions among `private`, `protected`, and `public` base class members in C++.

18. Describe the notion of selective availability in Eiffel.

19. How do the rules for member name visibility in Smalltalk and Objective-C differ from the rules of most object-oriented languages?

20. Howdo inner classes in Java differ from most nested classes?

21. Describe the key design difference between object-oriented features of Smalltalk, Eiffel, and C++, on the one hand, and Ada, CLOS, and Fortran on the other.

22. What are extension methods in C#? What purpose do they serve?

23. Does a constructor allocate space for an object? Explain.

24. What is a metaclass in Smalltalk?

25. Why is object initialization simpler in a language with a reference model of variables (as opposed t a value model)?

26. How does a C++ (or Java or C#) compiler tell which constructor to use for a given object? How does the answer differ for Eiffel and Smalltalk?

27. What is escape analysis? Describe why it might be useful in a language with a reference model of variables.

28. Summarize the rules in C++ that determine the order in which constructors are called for a class, its base class(es), and the classes of its fields. How are these rules simplified in other languages?

29. Explain the difference between initialization and assignment in C++.

30. Why does C++ need destructors more than Eiffel does?

31. Explain the difference between dynamic and static method binding (i.e., between virtual and nonvirtual methods).

32. Summarize the fundamental argument for dynamic method binding. Why do C++ and C# use static binding by default?

33. Explain the distinction between redefining and overriding a method.

34. What is a class-wide type in Ada 95?

35. Explain the connection between dynamic method binding and polymorphism.

36. What is an abstract method (also called a pure virtual method) in C++ and a deferred feature in Eiffel)?

37. What is a reverse assignment? Why does it require a run-time check?

38. What is a vtable? How is it used?

39. What is the fragile base class problem?

40. What is an abstract (deferred) class?

41. Explain the importance of virtual methods for object closures.

42. What is mix-in inheritance? What problem does it solve?

43. Outline a possible implementation of mix-in inheritance for a language with statically typed objects. Explain in particular the need for interface-specific views of an object.

44. Describe how mix-ins (and their implementation) can be extended with default method implementations, static (constant) fields, and even mutable fields.

45. What does true multiple inheritance make possible that mix-in inheritance does not?

46. What is repeated inheritance? What is the distinction between replicated and shared repeated inheritance?

47. What does it mean for a language to provide a uniform object mode? Name two languages that do so.

48. Given a few examples of the semantic ambiguities that arise when a class has more than one base class.

49. Explain the distinction between replicated and shared multiple inheritance. When is each desirable?

50. Explain how even nonrepeated multiple inheritance introduces the need for "`this` correction" fields in individual vtable entries.

51. Explain how shared multiple inheritance introduces the need for an additional level of indirection when accessing fields of certain parent classes.

52. Explain why true multiple inheritance is harder to implement than mix-in inheritance.

53. Name the three projects at Xerox PARC in the 1970s that pioneered modern GUI-based personal computers.

54. Explain the concept of a message in Smalltalk.

55. How does Smalltalk indicate multiple message arguments?

56. What is a block in Smalltalk? What mechanism does it resemble in Lisp?

57. Give three examples of how Smalltalk models control flow as message evaluation.

58. Explain how type checking works in Smalltalk.