Fragekatalog zur ersten AP-Schulaufgabe

1. Was versteht man unter einer Referenz in C++?
   Eine Referenz ist eine Art Zeiger auf eine Variable
2. Erklären Sie die Unterschiede zwischen einem C++ Pointer und einer C++ Referenz
   Ein Pointer (Zeiger) sollte zuletzt auf NULL gesetzt werden. Referenzen können nur einmal initialisiert werden.
3. Weshalb dürfen Referenzen nicht auf NULL gesetzt werden?
   Weil sie nur einmal initialisiert werden können. Sie wären also unveränderlich immer NULL. Sinnlos!
4. Erklären Sie auf welche Weise Funktionen in C++ „indirekt" mehrere Werte gleichzeitig zurückgeben können!
   Durch die Übergabe von Referenzen als Parameter an eine Methode
5. Was versteht man in C++ unter einem Funktionstemplate?
   Unter einem Funktionstemplate versteht man eine Funktion, die für abstrakte Datentypen als Generierungsparameter definiert ist.
6. Erläutern Sie Probleme bei der Nutzung von Funktionstemplates!
   Es gibt Probleme bei der Homonymauflösung
7. Beschreiben Sie die Vorteile von C++ Funktionstemplates!
   Methoden müssen nicht für jeden Datentyp einzeln implementiert werden, sie bleiben einfacher Wartbar und der Umfang des Codes (LOC) verringert sich. Außerdem findet eine charakteristische Typprüfung statt.
8. Erstellen Sie ein Funktionstemplate zu folgender Problembeschreibung...
   …
9. Aus welchen drei Teilen besteht die ANSI C++ Standard Template Library?
   - Container
   - Iteratoren
   - Algorithmen
10. Erklären Sie den Zweck von STL Algorithmen / Containern / Iteratoren
    - Container: Behälter für Programmobjekte, dynamisch erweiterbar
    - Iteratoren: abstrakte Zeiger auf Elemente und Container
    - Algorithmen: Folge von Abarbeitungsbefehlen zur Arbeit mit den Elementen im Container. (z.B. suchen, einfügen, ersetzen, löschen, …)
11. Nennen Sie die Arten von Containern, welche in der STL implementiert sind!
    - Listen
    - Sets
    - Maps
    - Adapter (Queue, Stack)

12. Beschreiben Sie die grundsätzliche Funktion der in C++ implementierten Container List, Set, Map, Queue und / oder Stack

- List: Geordnetes Speichern von Elementen in einer Liste
- Set: Doppelte Einträge sind verboten. Elemente werden als Schlüssel gespeichert
- Map: Speicherung als Schlüssel-Daten-Paar.
- Queue: FIFO- Prinzip. Elemente, die zuerst gespeichert wurden, werden zuerst herausgenommen
- Stack: FILO- Prinzip. Elemente, die zuerst gespeichert wurden, werden zuletzt herausgenommen

13. Definieren Sie z.B. einen list / vector – Container, der Zeiger auf double Werte speichern kann! Fügen Sie dem Container zwei Werte am Anfang hinzu!

…

14. Lesen Sie alle Werte aus einem Container über einen Iterator aus!

…

15. Erläutern Sie die Begriffe Kapselung, Verbergen von Daten, Vererbung und Polymorphie anhand eines Beispiels!

- Kapselung: Verbergen von Daten, Beschränkung des Zugriffs
- Vererbung: Spezialisierung, es entsteht eine „Ist-ein-Beziehung"
- Polymorphie: Verwendung von gleichnamigen Methoden in verschiedenen Kontexten

16. Erklären Sie den Unterschied zwischen Klasse und Objekt!

Eine Klasse definiert einen Bauplan für ein davon initiiertes Objekt. Ein Objekt ist eine Instanz einer Klasse.

17. Was versteht man unter

a) einem Attribut?

Die Eigenschaft einer Klasse bzw. eines Objekts

b) einer Methode?

Eine Funktion einer Klasse bzw. eines Objekts. Kann beliebig oft aufgerufen werden.

c) einer Instanz?

Initiierung einer Klasse.

18. Welche Eigenschaften und Methoden kann z.B. eine Klasse Bankkonto besitzen? (je drei Eigenschaften und Methoden)

- Eigenschaften:
  - Kontonummer
  - Guthaben
  - Kontoinhaber
- Methoden
  - GeldAuszahlen()
  - KontostandAnzeigen()
  - GeldEinzahlen()

19. Nennen Sie vier Zyklen eines objektorientierten Projekts!

- Analyse

- Design
- Realisierung
- Einsatz

20. Beschreiben Sie den Ablauf eines objektorientiert durchgeführten Projekts

Zunächst muss die Problemstellung analysiert werden. Der Entwickler muss erfahren, was der Auftraggeber eigentlich will. Anschließend muss das Design der Software erstellt werden. Danach erfolgt die Realisierung mit der Implementierung. Beim Einsatz der neuen Software müssen die Anwender geschult werden und eventuell auftretende Bugs dokumentiert werden. Die vier Phasen beginnen erneut. Dieses Vorgang wiederholt sich immer wieder.

21. Beschreiben Sie die Prozesse bei der Analyse eines objektorientierten Projekts

Eine Ist-Soll-Analyse muss durchgeführt werden. Im einem Lastenheft wird festgehalten, was der Auftraggeber will. Das Lastenheft gilt als eine Art Vertrag zwischen Auftraggeber und Entwickler. Das Lastenheft wird in der Regel immer wieder angepasst.

22. Erklären Sie den Unterschied zwischen Basis-, statischem und dynamischen Modell beim Design eines objektorientierten Projekts!

Das Basismodell beschreibt, welche Klassen benötigt werden.
Im statischen Modell werden die Beziehungen zwischen den Klassen festgehalten.
Das dynamische Modell beschreibt die Beziehungen zur Laufzeit.

23. Erläutern Sie die Prozesse bei der Realisierung eines objektorientierten Projekts!

- Kodierung: Sollte in einer objektorientierten Sprache erfolgen, da der Entwurfsansatz so direkt programmiertechnisch umsetzbar ist
- Test: Kontrolle, ob Ergebnisse der Designphase entsprechend umgesetzt wurden.

24. Beschreiben Sie die Abläufe beim Einsatz eines objektorientierten Projekts

- *Installation* auf produktiver Umgebung
- *Schulung* der Anwender
- *Review* zum Vergleich des Endprodukts mit dem geplanten Produkt.

25. Erklären Sie, weshalb die Zyklen bei der OOP mehrmals durchlaufen werden

Nach jedem Durchlauf aller Phasen nähert man sich dem Endprodukt näher. Erst wenn dieses erreicht ist, müssen die Zyklen nicht mehr durchlaufen werden.

26. Welche Vorteile bringt das Spiralmodell mit sich?

Man erhält relativ schnell eine erste Version des Produkts. Beim Einsatz kann der Kunde unmittelbar prüfen, ob alles zu seiner Zufriedenheit implementiert wurde. Sollte dies nicht der Fall sein, gibt er dies an die Entwickler weiter. Entwickler und Kunde arbeiten eng zusammen.

27. Warum wird bei der OOP auf das Design besonderes Augenmerk gelegt

Nach dem Design ist die größte Denkarbeit schon geschafft. Die UML stellt in der OOP die reale Welt einfach dar. Sie muss dann „nur" noch implementiert werden.

28. Vergleichen Sie den Entwurf eines strukturierten Softwareprojekts mit dem eines objektorientierten Objekts

???

29. Was versteht man unter einem Anwendungsfall?

Ein Anwendungsfall stellt ein System aus der Perspektive des Kunden her.

30. Beschreiben Sie die Symbolik des Anwendungsfallmodells!

Strichmännchen symbolisieren die Akteure. In einem System werden die Anwendungsfälle beschrieben. Striche stehen für die Beziehungen zwischen Akteuren und Anwendungsfällen.

31. Erläutern Sie die Bedeutung von Akteuren!

Akteure sind die Anwender der Systems. Dabei handelt es sich um Hardware aus Sicht des Kunden und um Objekte aus Sicht des Entwicklers.

32. Wozu benötigt man die <<include>>-Beziehung bei Anwendungsfallmodellen

Includes werden verwendet, wenn ein Anwendungsfall einen anderen Anwendungsfall enthält.

33. Beschreiben Sie den Unterschied zwischen der <<include>>- und der <<extend>>-Beziehung!

Beim Include enthält ein Anwendungsfall einen anderen Anwendungsfall. Beim Extend wird ein Anwendungsfall durch einen anderen Anwendungsfall erweitert.

34. Erklären Sie den Begriff „extension point"!

Genau in diesem Punkt wird ein Anwendungsfall durch einen anderen erweitert.

35. Vergleichen Sie das Konzept der Vererbung bei Klassen, mit dem bei Anwendungsfällen

Die Vererbungsbeziehung ist bei Klassen und Anwendungsfällen die gleiche.

36. Welche Vorteile bietet die Darstellung von Anwendungsfällen beim Softwareentwurf?

Sie stellt das System aus der Sicht des Kunden dar. Bei Gesprächen zwischen Entwickler und Kunden ist diese Darstellung sehr vorteilhaft.

37. Erstellen Sie ein Anwendungsfallmodell für folgende Problembeschreibung …

…

38. Erläutern Sie die Aussage des folgenden Klassendiagramms / Objektdiagramms nach UML! …

…

39. Beschreiben Sie die Beziehung zwischen Klassen, die durch Vererbung entsteht

Es entsteht eine „Ist-ein-Beziehung". Ein Objekt der Subklasse ist ein Objekt der Oberklasse. Das Objekt der Subklasse ist eine Spezialisierung.

40. Was versteht man unter einer abstrakten Klasse?
Eine Klasse, in der nicht alle Methoden vollständig implementiert sind. Mindestens eine Methode muss abstrakt sein. Das heißt, dass diese von einer Subklasse implementiert werden muss.

41. Erklären Sie den Unterschied zwischen einer Interface-Klasse und einer abstrakten Klasse
Klassen, die ein Interface implementieren, müssen alle Methoden des Interfaces überschreiben.
Abstrakte Klassen implementieren manche Methoden aus, mindestens eine Methode ist abstrakt.
Abstrakte Klassen können initiiert werden, Interfaces nicht.

42. Wie werden abstrakte Klassen/ Interface Klassen in C++ erstellt
In Interfaces werden ALLE Methoden als virtual deklariert. In abstrakten Klassen wird mindestens eine Methode als virtual deklariert.

43. Nennen Sie … mögliche Assoziationen aus dem objektorientierten Entwurf und geben Sie je ein Beispiel dazu an!
…

44. Erklären Sie den Unterschied zwischen einer Aggregation und einer Komposition!
- Aggregation: Ist-Teil-Von-Beziehung. Eine Klasse ist Teil einer anderen Klasse. Sie kann aber unabhängig von der anderen Klasse existieren.
- Komposition: Verwendet-Beziehung. Eine Klasse ist Teil einer anderen Klasse. Sie kann aber nicht existieren, wenn die andere Klasse nicht existiert.

45. Was versteht man unter einer Abhängigkeit?
Eine „Wer-kennt-wen"-Beziehung. Die abhängige Klasse kennt die Unabhängige. Die Unabhängige jedoch nicht die Abhängige. Abhängigkeiten gibt es nicht nur zwischen Klassen.

46. Erläutern Sie die <<use>>-Abhängigkeit zwischen zwei Klassen anhand eines Beispiels!
Die abhängige Klasse verwendet die unabhängige Klasse. Beispiel: Ein Thin-Client verwendet die Prozessorleistung des Servers.

47. Beschreiben Sie die Rollen von Client und ein Supplier bei der Abhängigkeitsbeziehung
Der Client beschreibt die abhängige Klasse. Er ist abhängig vom Supplier, der unabhängigen Klasse.

48. Was versteht man unter einer Assoziationsklasse?
Eine Assoziationsklasse beschreibt die Beziehung zwischen zwei Klassen.

49. Zeichnen Sie ein Klassendiagramm / Objektdiagramm nach UML, das die folgende Situation beschreibt …
…

50. Erläutern Sie die Unterschiede der Schutzattribute public, protected und private!
- Public: Attribut ist für alle sichtbar, kann von jeder anderen Klasse verwendet und geändert werden.

- Protected: Attribut kann nur von abgeleiteten Klassen verwendet werden.
- Private: Attribut kann nur innerhalb der eigenen Klasse verwendet werden.

51. Implementieren Sie die Klassendefinitionen in C++ zu folgenden Klassendiagrammen nach UML …

…

52. Welchem Zweck dient ein Zustandsdiagramm / Aktivitätsdiagramm / Kommunikationsdiagramm / Sequenzdiagramm / Timingdiagramm?

- Ein Zustandsdiagramm beschreibt die Zustände, die ein Objekt annehmen kann, sowie die Übergänge zwischen den Zuständen.
- Ein Aktivitätsdiagramm beschreibt die Aktivitäten eines Systems in Abhängigkeit von den Zuständen.
- Ein Kommunikationsdiagramm beschreibt die Zusammenarbeit zwischen den Objekten.
- Ein Sequenzdiagramm beschreibt den zeitlichen Ablauf eines Systems sowie den Nachrichtenverlauf zwischen den Objekten.
- Timingdiagramme zeigen den zeitlichen Kontext, in dem sich die Zustände der beteiligten Objekte ändern.

53. Weshalb verwendet man in der objektorientierten Programmierung keine Struktogramme zur Darstellung des Programmablaufs?

Struktogramme arbeiten nicht objektorientiert sondern sequenziell.

54. Worin unterscheiden sich Kommunikationsdiagramme von Sequenzdiagrammen?

Im Kommunikationsdiagramm sind die Nachrichten zwischen den Objekten durchnummeriert. Im Sequenzdiagramm wird die Lebensdauer der beteiligten Objekte genauer beschrieben.

55. Was versteht man unter dem „Tokenprinzip“ bei Aktivitätsdiagrammen?

Es kann immer nur ein Pfad mit einem Token durchlaufen werden.

56. Erläutern Sie das Konzept der Partitionen beim Aktivitätsdiagramm

Innerhalb von Partitionen wird dargestellt, wer für welche Aktion verantwortlich ist.

57. Welche Diagramme der UML zählen zu den Interaktionsdiagrammen?

- Sequenzdiagramme
- Kommunikationsdiagramme
- Timingdiagramme

58. Unter welchen Umständen setzt man Interaktionsübersichtsdiagramme in der UML ein?

Zur Darstellung von größeren Zusammenhängen

59. Erläutern Sie die Bedeutung der dynamischen Modellierung für die objektorientierte Programmentwicklung!

Statische Modelle können nur Beziehungen zwischen Klassen darstellen. Dynamische Modelle beschreiben Beziehungen zur Laufzeit.

60. Erstellen Sie ein Zustandsdiagramm / Aktivitätsdiagramm / Kommunikationsdiagramm / Sequenzdiagramm zu folgender Problembeschreibung …

…

61. Erstellen Sie den C++ Quelltext zu folgendem Zustandsdiagramm / Aktivitätsdiagramm / Kollaborationsdiagramm / Sequenzdiagramm zu folgender Beschreibung …

…

62. Nennen Sie die beiden UML Modellelemente des Komponentendiagramms

Komponenten und Artefakte

63. Erklären Sie den Unterschied zwischen einer Komponente und einem Artefakt!

Komponenten stellen einen modularen Teil eines Systems dar.
Artefakte sind physische Informationseinheiten.

64. Erläutern Sie drei gebräuchliche Stereotypen von Artefakten!

- File
- document
- executable

65. Was versteht man unter einem „Subsystem"?

Ein Teilsystem des übergeordneten Systems.

66. Auf welche Weise werden Schnittstellen symbolisiert?

Durch einen Kreis.

67. Erläutern Sie die Bedeutung von Komponenten hinsichtlich der Wiederverwendbarkeit von Software!

Über Schnittstellen können Komponenten auch von anderen Systemen verwendet werden.

68. Erstellen Sie ein Komponentendiagramm / Verteilungsdiagramm zu folgender Problembeschreibung …

…

Viel Erfolg beim Lernen!