Zusammenfassung M320 code prüfung

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte BeschreibungKlassendiagramm

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Webseite enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Software, Webseite enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Software, Webseite enthält.

Automatisch generierte Beschreibung

1. **Nutzen von Assoziationen im Rahmen von objektorientierten Programmen:**
   * Dieses Lernziel befasst sich damit, wie Objekte in einem Programm miteinander in Beziehung stehen können. Assoziationen repräsentieren diese Beziehungen zwischen Objekten. Zum Beispiel kann ein **Student**-Objekt eine Assoziation zu einem **Kurs**-Objekt haben, um anzuzeigen, dass der Student an diesem Kurs teilnimmt.
2. **Arten von Assoziationen (Assoziation, Aggregation, Komposition):**
   * Assoziation: Eine lose Beziehung zwischen Objekten. Beispiel: Ein **Lehrer** ist mit einer **Schule** assoziiert.
   * Aggregation: Eine spezielle Assoziation, bei der ein Objekt ein Teil eines anderen Objekts ist, aber unabhängig existieren kann. Beispiel: Ein **Auto** hat einen **Motor**.
   * Komposition: Eine enge Beziehung, bei der ein Objekt Teil eines anderen ist und ohne das Hauptobjekt nicht existieren kann. Beispiel: Ein **Haus** besteht aus **Zimmern**.
3. **Generalisierung, Vererbung, Superklasse, Subklasse:**
   * Generalisierung bezieht sich auf das Erstellen von Klassen-Hierarchien, bei denen Subklassen von Superklassen abgeleitet werden. Vererbung ermöglicht das Teilen von Eigenschaften und Verhalten von Superklassen in Subklassen.
   * Superklasse (Base Class): Die Oberklasse in einer Vererbungshierarchie. Beispiel: **Fahrzeug** als Superklasse für **Auto** und **Motorrad**.
   * Subklasse (Derived Class): Eine Klasse, die von einer Superklasse erbt. Beispiel: **Auto** als Subklasse von **Fahrzeug**.
4. **Vererbung als Instrument zur Abstraktion:**
   * Vererbung ermöglicht die Wiederverwendung von Code und die Bildung von abstrakten Superklassen, die gemeinsame Eigenschaften und Verhalten definieren. Dies fördert die Abstraktion und Strukturierung des Codes.
5. **Membervererbung und Konsequenzen:**
   * Bei der Vererbung in C# werden Member (Methoden, Eigenschaften, Felder) von der Super- an die Subklasse vererbt. Die Konsequenzen betreffen die Sichtbarkeit und Verwendung dieser Member in Subklassen. Zum Beispiel werden **public** Member vererbt, während **private** Member nicht sichtbar sind.
6. **Zugriffsmodifizierer (public, private, protected) in Vererbung:**
   * Zugriffsmodifizierer regeln die Sichtbarkeit von Membern. **public** Member sind in allen Klassen sichtbar, **private** Member nur in der Klasse, die sie definiert, und **protected** Member in abgeleiteten Klassen. Dies beeinflusst die Kapselung.
7. **Syntax der Generalisierung in C#:**
   * In C# wird die Generalisierung (Vererbung) durch das Schlüsselwort **:**, gefolgt von der Superklasse, in der Klassendefinition angegeben. Beispiel: **class Subklasse : Superklasse**.
8. **Überschreibung von Methoden und Instanzvariablen in C#:**
   * In C# können Methoden und Instanzvariablen mit den Schlüsselwörtern **virtual** und **override** überschrieben werden, um das Verhalten in Subklassen anzupassen.
9. **Polymorphie und Anwendungsfälle:**
   * Polymorphie ermöglicht es, auf unterschiedliche Objekte in einer konsistenten Weise zuzugreifen. Zum Beispiel kann eine Methode, die auf einer Superklasse basiert, mit verschiedenen Subklassenobjekten aufgerufen werden.
10. **Steuerungsumkehr (Inversion of Control) und Vorteile:**
    * Inversion of Control ist ein Designprinzip, bei dem die Kontrolle über die Ausführung von Code an ein Framework oder eine Containerklasse übergeben wird. Dies fördert die Lockerung von Abhängigkeiten und Testbarkeit.
11. **Techniken zur Implementierung der Steuerungsumkehr in C#:**
    * Es gibt verschiedene Techniken, wie Dependency Injection, Factory-Muster und Service Locator, um IoC in C# umzusetzen.
12. **Dependency Injection und Konstruktor-Injektion:**
    * Dependency Injection (DI) ist eine Technik, bei der Abhängigkeiten in eine Klasse injiziert werden, anstatt dass die Klasse sie selbst erstellt. Konstruktor-Injektion ist eine Form der DI, bei der Abhängigkeiten über den Konstruktor übergeben werden.
13. **Nutzen von Dependency Injection für Komponententests:**
    * Dependency Injection erleichtert das Testen von Komponenten, da Abhängigkeiten durch Mock-Objekte oder Testdoubles ersetzt werden können, um isolierte Tests durchzuführen.
14. **Optimierung von produktivem Code mittels Dependency Injection für Komponententests:**
    * Dependency Injection ermöglicht die Trennung von Code für die Geschäftslogik von Code für die Verwaltung von Abhängigkeiten, was die Wartbarkeit und Testbarkeit verbessert.

Konstruktoren erstellen eine Person können Parameter mitgeben

class Person

{

public string Name { get; set; }

public int Age { get; set; }

// Konstruktor, der den Namen setzt und das Alter auf 0 initialisiert

public Person(string name) : this(name, 0)

{

}

// Hauptkonstruktor, der Name und Alter setzt

public Person(string name, int age)

{

Name = name;

Age = age;

}

Konstruktor geben Parameter weiter die ein Objekt nutzen kann eine Methode braucht Parameter, deren Wert sie dann für die eigene Methode verwendet.

**Dynamische Arrays (Liste):**

* Die **List<T>**-Klasse in C# bietet eine flexible Möglichkeit, dynamische Arrays zu erstellen, die automatisch erweitert werden können.

csharpCopy code

List<int> dynamicArray = new List<int>();

dynamicArray.Add(1);   
dynamicArray.Add(2);   
dynamicArray.Add(3);  
**Foreach**

string[] namen = { "Alice", "Bob", "Charlie" };

foreach (string name in namen)

{  
 Console.WriteLine(name);  
}