🎓 Java OOP & Autovermietung – Multiple Choice Quiz

📝 Fragen

1. Was beschreibt eine abstrakte Klasse korrekt?

- ☐ Eine Klasse ohne Konstruktor
- ☐ Eine Klasse, die nicht instanziiert werden kann
- ☐ Eine Klasse ohne Attribute
- ☐ Eine Klasse ohne Methoden

2. Welche Aussage über Interfaces ist korrekt?

- ☐ Interfaces können private Attribute haben
- ☐ Interfaces können Konstruktoren haben
- ☐ Interfaces können keine Methoden enthalten
- ☐ Interfaces können `static` Methoden enthalten

3. Was bewirkt das Schlüsselwort `extends`?

- ☐ Eine Klasse implementiert ein Interface
- ☐ Eine Klasse instanziiert ein Objekt
- ☐ Eine Klasse erbt von einer anderen Klasse
- ☐ Eine Klasse wird zu einem Interface

4. Wie viele abstrakte Methoden darf ein Interface in Java haben?

- ☐ Keine
- ☐ Genau eine
- ☐ Beliebig viele
- ☐ Nur zwei

5. Welcher Modifizierer steht in UML für `protected`?

- ☐ +
- ☐ -
- ☐ #
- ☐ *

6. Was ist richtig über `ArrayList`?

- ☐ Sie ist eine dynamisch wachsende Liste
- ☐ Sie ist synchronisiert
- ☐ Sie hat eine feste Größe
- ☐ Sie kann nur Strings speichern

7. Welche Methoden müssen bei einer anonymen Klasse überschrieben werden?

- [ ] Alle öffentlichen Methoden der Object-Klasse
- [ ] Die Methoden des implementierten Interfaces oder der erweiterten abstrakten Klasse
- [ ] Keine Methoden
- [ ] Nur private Methoden

8. Welche Klassenstruktur entspricht dem Autovermietungsbeispiel?

- [ ] Interface → Abstrakte Klasse → Konkrete Klasse
- [ ] Interface → Konkrete Klasse → Abstrakte Klasse
- [ ] Abstrakte Klasse → Interface → Konkrete Klasse
- [ ] Konkrete Klasse → Interface → Abstrakte Klasse

9. Was trifft auf eine `private` Methode zu?

- [ ] Sie ist nur innerhalb der eigenen Klasse sichtbar
- [ ] Sie ist überall sichtbar
- [ ] Sie kann in Unterklassen überschrieben werden
- [ ] Sie muss `@Override` verwenden

10. Warum verwendet man `@Override`?

- [ ] Um anzuzeigen, dass eine Methode überschrieben wird
- [ ] Um neue Attribute zu definieren
- [ ] Um mehrere Konstruktoren zu deklarieren
- [ ] Um Variablen `final` zu machen

11. Welche Kombination von Schlüsselwörtern ermöglicht Mehrfachvererbung?

- [ ] `extends` und `extends`
- [ ] `extends` und `implements`
- [ ] Nur `extends`
- [ ] Nur `implements`

12. Wie kann man die Methode `berechnePreis()` sinnvoll erweitern?

- [ ] Mit einer Schleife
- [ ] Mit zusätzlichen Gebühren (z.B. Versicherung) in der Berechnung
- [ ] Mit einem Array von Strings
- [ ] Durch Entfernen von Parametern

13. Was bewirkt Polymorphismus im Autovermietungsbeispiel?

- [ ] Mehrere Klassen erben von einem Interface, aber sind inkompatibel
- [ ] Gleicher Typ (z.B. Fahrzeug), aber unterschiedliche Implementierungen
- [ ] Nur Interfaces können polymorph sein
- [ ] Nutzung von gemeinsamen Typen bei unterschiedlichen Implementierungen

14. Welche Aussage über Konstruktoren ist korrekt?

- [ ] Ein Konstruktor hat keinen Rückgabewert
- [ ] Ein Konstruktor ist immer `public`
- [ ] Konstruktoren dürfen nur in Interfaces sein
- [ ] Konstruktoren müssen `static` sein

15. Welche Codezeile erstellt eine anonyme Klasse korrekt?

- [ ] new Auto();
- [ ] new Fahrzeug();
- [ ] new Fahrzeug() { public String getMarke() { return "X"; } public String getModell() { return "Y"; } public double getTagespreis() { return 100; } };
- [ ] Auto a = Fahrzeug();

16. Was bedeutet `encapsulation` (Kapselung) in Java?

- [ ] Methoden haben den gleichen Namen aber unterschiedliche Parameter
- [ ] Attribute sind direkt öffentlich zugänglich
- [ ] Verbergen von Daten durch private Felder und öffentliche Methoden
- [ ] Alles im Code wird verschlüsselt

17. Wozu dient `super()` im Konstruktor?

- [ ] Es ruft den Konstruktor der Unterklasse auf
- [ ] Es ruft eine statische Methode auf
- [ ] Es ruft den Konstruktor der Oberklasse auf
- [ ] Es überschreibt Methoden automatisch

18. Wie ruft man in Java einen Konstruktor einer Elternklasse auf?

- [ ] this()
- [ ] super()
- [ ] parent()
- [ ] extends()

19. Wie kann man in Java ein Interface deklarieren?

- [ ] public interface Fahrzeug {}
- [ ] public abstract Fahrzeug {}
- [ ] public static Fahrzeug {}
- [ ] public interface Fahrzeug extends Klasse {}

20. Welche Typen von Schleifen gibt es in Java?

- [ ] for, while, until
- [ ] loop, do, while
- [ ] for, while, do-while
- [ ] repeat, until, while

21. Wie können Objekte von einer abstrakten Klasse erstellt werden?

- [ ] new AbstractClass()
- [ ] Mit Hilfe einer anonymen Klasse
- [ ] Direkt mit dem Interface
- [ ] Mit `abstract new()`

22. Welches Schlüsselwort nutzt man für Konstanten?

- [ ] static final
- [ ] constant
- [ ] immutable
- [ ] protected static

23. Wann wird `instanceof` verwendet?

- [ ] Zum Überprüfen der Vererbungshierarchie
- [ ] Um Objekte zu casten
- [ ] Um Konstruktoren aufzurufen
- [ ] Um statische Methoden zu überprüfen

24. Welche Vorteile hat eine `List` gegenüber einem Array?

- [ ] Sie hat eine feste Größe
- [ ] Sie kann nicht wachsen
- [ ] Sie ist dynamisch in der Größe
- [ ] Sie ist schneller als alle anderen Collections

25. Wie kann man Klassen in Paketen organisieren?

- [ ] Durch das Keyword `package`
- [ ] Mit `classgroup`
- [ ] Über `group`
- [ ] Mit `directory`

26. Was bedeutet das Prinzip der Vererbung?

- [ ] Klassen teilen sich Speicher
- [ ] Eine Klasse erbt Methoden und Attribute von einer anderen
- [ ] Alle Methoden werden überschrieben
- [ ] Methoden sind alle private

27. Wie kann man die Sichtbarkeit auf Klassenebene am stärksten beschränken?

- [ ] public
- [ ] private
- [ ] protected
- [ ] default (package-private)

28. Was bedeutet Method Overloading?

- [ ] Mehrere Methoden mit gleichem Namen und unterschiedlichen Parametern

- [ ] Eine Methode ruft sich selbst auf
- [ ] Methoden überschreiben Elternmethoden
- [ ] Methoden haben keine Parameter

29. Welche von folgenden ist KEIN Sammlungstyp in Java?

- [ ] List
- [ ] Map
- [ ] Set
- [ ] Table

30. Welche Aussage zu anonymen Klassen ist richtig?

- [ ] Sie haben einen Namen, werden aber versteckt
- [ ] Sie werden oft verwendet, wenn man nur einmalig eine Klasse benötigt
- [ ] Sie sind immer `static`
- [ ] Sie müssen mehrere Interfaces gleichzeitig implementieren