# Fülle den Lückentext aus (40 Punkte)

Ordne jeder Lücke im untenstehenden Lückentext einen Begriff zu, indem du die zugehörige Nummer des Begriffs in die entsprechende Lücke einträgst. Manche Wörter kommen mehrfach vor. Die hier aufgezählten Begriffe berücksichtigen weder Numerus noch Kasus.

1. Unterklasse
2. interagieren
3. public
4. Objekt-Fabrik
5. Zustand
6. Objektorientiert
7. Referenz
8. Objekt
9. Package-Private
10. Feld
11. Objekt-Bauplan
12. Methode
13. Delegation
14. Wert
15. Instanz
16. Aufgabe
17. Verhalten
18. Gleichartig
19. Individuell
20. Kontrollieren
21. This
22. Package
23. Zugriff
24. Vererbend
25. Identität
26. Zustand
27. Wert
28. Privat
29. Unterklasse
30. Implementieren
31. Zugriffsmodifizierer
32. Inhaltlich
33. Objektorientiert
34. Superklasse
35. Gemeinsam
36. Klasse
37. Oberklasse
38. Aufrufen
39. Paradigma
40. protected

Die **Objektorientierte** Programmierung ist ein **Paradigma** zum Entwerfen und **Implementieren** von Software. Bei diesem Ansatz wird ein Programm als Menge **interagierender** Objekte modelliert. Jedes Objekt übernimmt eine spezifische **Aufgabe** und trägt so zur Gesamtaufgabe der Applikation bei.

Ein Objekt besitzt einen individuellen **Zustand**, der auch als Gedächtnis des Objektes bezeichnet wird. Objekte haben ein **Verhalten**, können also etwas tun und damit ihren **Zustand** verändern. Jedes erzeugte Objekt besitzt eine eigene **Identität**. Selbst wenn zwei Objekte **inhaltlich** gleich sind, sind sie dennoch nicht dasselbe.

Um **gleichartige** Objekte zu erzeugen, verwendet man Klassen, auch **Objekt-Baupläne** oder **Objekt-Fabriken** genannt. Diese definieren, welche Attribute und Operationen die zu erstellenden Objekte – auch **Instanzen** genannt - besitzen. In Java werden Attribute als **Felder** und Operationen als **Methoden** implementiert.

In den meisten **objektorientierten** Programmiersprachen unterscheidet man zwischen Klassenvariablen und Instanzvariablen. Eine Klassenvariable speichert einen **Wert**, der von allen Objekten der Klasse **gemeinsam** verwendet wird. Instanzvariablen hingegen speichern **Werte**, die für **individuelle** Objekte gelten. Eine Klassenmethode wird direkt auf der **Klasse** aufgerufen und eine Instanzmethode muss immer im Kontext eines **Objektes** aufgerufen werden. Klassenmethoden haben keinen **Zugriff** auf Instanzvariablen. Umgekehrt kann eine Instanzmethode aber auf Klassenmethoden und Klassenvariablen zugreifen.

Um eine Instanzmethode **aufzurufen**, benötigt man eine **Referenz** auf ein Objekt. Soll eine Methode des eigenen Objekts ausgeführt werden, benutzt man das Schlüsselwort **this**. Dieses lässt sich auch für die sogenannte Konstruktor**-Delegation** verwenden.

Klassen können voneinander erben. Dabei stellt eine Klasse die **Oberklasse** und die andere die **Unterklasse** dar. Die erbende Klasse erhält dabei alle Felder und Methoden der **Superklasse**, kann auf diese aber nicht immer direkt zugreifen. Ein **privates** Feld ist für die erbende Klasse beispielsweise nicht sichtbar. Befindet sich die erbende Klasse in einem anderen **Paket** als die **vererbende** Klasse, so sind auch die geerbten **package-private** Felder nicht sichtbar. Der Zugriff auf **public** Felder ist hingegen immer möglich. Eine Sonderstellung nehmen die mit dem Schlüsselwort **protected** deklarierten Variablen ein. Diese sind in Java von allen Klassen im selben Paket erreichbar und zusätzlich von allen **Unterklassen**, selbst wenn sich diese in einem anderen Paket befinden. Die Schlüsselwörter, die man zum **Kontrollieren** der Sichtbarkeit bzw. Erreichbarkeit verwendet, nennt man auch **Zugriffsmodifizierer**.

# Löse das Kreuzworträtsel (25 Punkte)

Die folgenden Begriffe sind den darunter stehenden Beschreibungen zuzuordnen und in das Kreuzworträtsel auf der nächsten Seite einzutragen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instanzvariable | rekursiv | überschreiben | Referenz | static |
| final | Methode | polymorph | Paradigma | überladen |
| Assoziation | String | Komposition | Runtime | Integer |
| Schleife | new | Compiler | Instanz | Member |
| Array | Sequenz | Klassenvariable | Aggregation | Konstruktor |

1. Eine andere Bezeichnung für ein Objekt.
2. Eine Funktion innerhalb einer Klasse.
3. Eine Datenstruktur, die Elemente sequentiell speichert.
4. Ein Schlüsselwort, um Methode als Klassenmethode zu kennzeichnen.
5. Mit diesem Operator erzeugt man ein Objekt.
6. Dieser Datentyp repräsentiert eine ganze Zahl.
7. Dieser Datentyp repräsentiert eine Zeichenfolge.
8. Diese besondere Methode initialisiert ein neues Objekt.
9. Speichert einen Wert für ein Objekt.
10. Speichert einen gemeinsamen Wert für eine gesamte Klasse und deren Objekte.
11. Dieses Schlüsselwort kennzeichnet eine unveränderliche Variable.
12. Speichert einen Verweis auf ein Objekt.
13. Eine Kontrollstruktur zur Wiederholung von Anweisungen.
14. Eine Kontrollstruktur, die Anweisungen nacheinander ausführt.
15. Bezeichnet die Fähigkeit einer Variablen, Objekte unterschiedlichen Datentyps zu referenzieren und über dieselbe Schnittstelle anzusprechen.
16. Felder und Methoden sind ... einer Klasse.
17. So bezeichnet man eine Methode, die sich selbst aufruft.
18. So bezeichnet man eine Denkweise, eine Herangehensweise oder einen Ansatz.
19. Eine semantische Beziehung zwischen 2 Klassen.
20. Eine Beziehung zwischen Klassen, bei der die eine Klasse einen Teil der anderen darstellt.
21. Eine Beziehung zwischen Klassen, bei der das "Teil" existenziell abhängig vom "Ganzen" ist.
22. Einen zweiten Parametersatz für eine bereits existierende Methode definieren.
23. Eine Methode aus einer Oberklasse inhaltlich ergänzen oder inhaltlich ersetzen.
24. Ein Programm, das Quelltext übersetzt.
25. Ein Programm, das ein anderes Programm ausführt und Ressourcen bereitstellt.

