Your quiz solution has been submitted successfully.

Menu

Objektorientierte Programmierung in Java openHPI-Java-Team

Recap

Time effort: approx. 19 minutes

Question 1 2.0 / 2.0

Wähle aus den vorgegebenen Optionen die Objekte in folgendem Codebeispiel aus:

```
class Story {
    public static void main(String[] args) {
        Robot robin = new Robot();
        robin.sayAge();
        Task task = new Task();
        System.out.println(task.name);
    }
}
```

Hide Explanation

Objekte werden aus Klassen instanziiert. Sie werden klein geschrieben, um sie im Code leicht von den Klassen unterscheiden zu können.
Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.2 Klassen und Objekte an und wiederhole die Übungsaufgaben dazu.

- [ ] Robot
- [x] robin — Correct!
- [ ] Robot()
- [ ] sayAge()
- [ ] new
- [ ] Task
- [x] task — Correct!
- [ ] Task()
- [ ] name

Question 2 1.5 / 2.0

Wähle die Aussagen aus, die wahr sind.

Hide Explanation

States beschreiben den Zustand eines Objektes. Der Zustand wird durch Attribute beschrieben. Behaviour hingegen beschreibt das Verhalten eines Objekts. Das Verhalten wird durch Methoden beschrieben.

- [ ] Der Zustand eines Objekts wird durch Methoden beschrieben
- [x] Behaviour beschreibt das Verhalten eines Objekts — Correct!
- [x] Der Zustand eines Objekts wird durch Attribute beschrieben — Correct!
- [ ] Behaviour beschreibt den Zustand eines Objekts
- [x] State beschreibt den Zustand eines Objekts — Correct!
- [ ] Das Verhalten eines Objekts wird durch Attribute beschrieben

- [x] Das Verhalten eines Objekts wird durch Methoden beschrieben — Correct!
- [x] State beschreibt das Verhalten eines Objekts — Your Answer

Question 3 2.0 / 2.0

Wähle aus den vorgegebenen Optionen die Klassenbezeichner in folgendem Codebeispiel aus:

Datei Robot:

```
class Robot {
    int age = 1;
    void sayAge() {
        System.out.println(age);
    }
}
```

Datei Task:

```
class Task {
    String name = "Finde Paco!!";
}
```

Datei Story:

```
class Story {
    public static void main(String[] args) {
        Robot robin = new Robot();
        robin.sayAge();
        Task task = new Task();
        System.out.println(task.name);
    }
}
```

Hide Explanation

Klassen sind die Baupläne für konkrete Objekte. Sie werden mit dem Schlüsselwort class beschrieben.
Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.2 Klassen und Objekte an und wiederhole die Übungsaufgaben dazu.

- [ ] Robot()
- [x] *in Datei Story:* Story — Correct!
- [ ] task
- [ ] new
- [x] *in Datei Robot:* Robot — Correct!
- [ ] sayAge()
- [ ] robin
- [ ] Task()
- [ ] name
- [x] *in Datei Task:* Task — Correct!

Question 4 2.0 / 2.0

Wähle die Aussagen aus, die auf folgendes Codebeispiel zutreffen.

```
class Robot {
    int age = 1;

    void printRoboInfo() {
        String name = "Robin";
        System.out.println(name + " ist " + age + " Jahr alt.");
    }
}
```

Hide Explanation

In Java gibt es sowohl Attribute als auch Variablen. Attribute bilden den Zustand von Objekten ab und werden in einer Klasse definiert. Variablen werden in Methoden definiert und bilden nicht den Zustand von Objekten ab. Diese werden zum Beispiel für kleinere Berechnungen, Zwischenspeicherung von Ergebnissen etc. genutzt.

- [x] age ist ein Attribut — Correct!
- [ ] age ist ein Methodenaufrauf
- [ ] age ist eine Methode
- [ ] age ist eine Variable
- [ ] name ist ein Attribut
- [ ] name ist ein Methodenaufruf
- [ ] name ist eine Methode
- [x] name ist eine Variable — Correct!

Question 5 2.0 / 2.0

Welche dieser Aussagen zum nachfolgenden Codebeispiel ist korrekt?

```
String name = "Robin";
```

Hide Explanation

Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.3 Variablen (1) an und wiederhole die Übungsaufgaben dazu.

- ( ) Robin ist der Datentyp, String der Bezeichner und name der Wert
- ( ) name ist der Datentyp, String der Bezeichner und Robin der Wert
- ( ) Robin ist der Datentyp, name der Bezeichner und String der Wert
- ( ) name ist der Datentyp, Robin der Bezeichner und String der Wert
- (•) String ist der Datentyp, name der Bezeichner und Robin der Wert — Correct!
- ( ) String ist der Datentyp, Robin der Bezeichner und name der Wert

Question 6 2.0 / 2.0

Ist dieses Codebeispiel korrekt?

```
void printName() {
    String name = "Kein Name";
    String name = "Robin";
    System.out.println(String name);
}
```

Hide Explanation

Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.3 Variablen (1) an und wiederhole die Übungsaufgaben dazu.

- [x] Nein, weil der Aufruf System.out.println(String name) falsch ist — Correct!
- [x] Nein, weil die Variable *name* mehrmals in der Methode printName() angelegt wird — Correct!
- [ ] Ja
- [ ] Nein, weil der Wert einer einmal angelegten Variable nicht mehr verändert werden darf

Question 7 2.0 / 2.0

Welche dieser Operatoren gibt es in Java?

Hide Explanation

Es gibt unter anderem die Operatoren +=, -=, ++, --, *= und /= in Java. Die Operatoren, die ein = enthalten, führen zunächst die Rechenoperation links vom Gleichheitszeichen (also +, -, * oder /) aus und weisen den Wert dann der Variablen zu. Beispiel:

```
int i = 2; // i wird auf 2 gesetzt
i *= 3; // i ist nun 6
```

Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.4 Variablen (2) an.

- [x] *= — Correct!
- [ ] =/
- [ ] =-
- [x] ++ — Correct!
- [x] += — Correct!
- [x] -- — Correct!

Question 8 2.0 / 2.0

Wähle aus den vorgegebenen Optionen die Methodenaufrufe in folgendem Codebeispiel aus:

```
class Story {
    public static void main(String[] args) {
        Robot robin = new Robot();
        robin.sayAge();
        Task task = new Task();
        System.out.println(task.name);
    }
}
```

Hide Explanation

Beim Methodenaufruf wird der Punkt-Operator verwendet, um auf die Methode des Objekts (und später im Kursverlauf auch auf die Methode einer Klasse) zuzugreifen. Ein Methodenaufruf beinhaltet immer runde Klammern, so dass der Befehl `robin.sayAge();` die Methode **sayAge()** des Objekts robin aufruft. **sayAge()** ist hier also der Methodenaufruf. Durch die Klammer lässt sich die Methode, also das Verhalten des Objekts, eindeutig von den Attributen, also dem Zustand des Objekts, unterscheiden.
Was Robot() und Task() ist, lernen wir am Ende der zweiten Woche.
Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.6 Methoden an und wiederhole die Übungsaufgaben dazu.

- [ ] Task
- [ ] task
- [ ] name
- [ ] Robot
- [ ] new
- [ ] robin
- [x] sayAge() Correct!

Question 9 2.0 / 2.0

Wähle aus den vorgegebenen Optionen die Attributzugriffe in folgendem Codebeispiel aus:

```
class Story {
    public static void main(String[] args) {
        Robot robin = new Robot();
        robin.sayAge();
        Task task = new Task();
        System.out.println(task.name);
    }
}
```

Hide Explanation

Beim Attributzugriff wird der Punkt-Operator verwendet, um auf das Attribut des Objekts (und später im Kursverlauf auch auf die Attribute einer Klasse) zuzugreifen. Ein Attributzugriff beinhaltet **nie** runde Klammern, so dass der Befehl `task.name;` das Attribut *name* des Objekts task aufruft. *name* ist hier also der Attributzugriff.
Durch die fehlende Klammer lassen sich Attribute, also der Zustand des Objekts, eindeutig von den Methoden, also dem Verhalten des Objekts, unterscheiden.
Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.5 Attribute an und wiederhole die Übungsaufgaben dazu.

- [ ] sayAge()
- [ ] task
- [ ] Task
- [ ] Robot
- [x] name Correct!
- [ ] Task()
- [ ] robin
- [ ] new
- [ ] Robot()

Question 10 2.0 / 2.0

Wähle die Aussagen aus, die bei folgendem Codebeispiel wahr sind:

```
class Robot {
   double a = 4;
   double b = 2.5;
   String info = "Ich bin ein Roboter und so alt: ";
   int age = 2;

    void printName() {
        String name = "Robin";
        return name;
    }

    int calculate() {
        return a-b;
    }

    String printInfo() {
        return info + age;
    }
}
```

Hide Explanation

Das Codebeispiel erzeugt Fehler in den Methoden **printName()** und **calculate()**.
Der Rückgabewert von **printName()** ist void, also der leere Rückgabetyp. Daher darf hier return nicht verwendet werden. Statt des returns könnte hier System.out.println(name) verwendet werden. Soll der Name zurückgegeben werden, muss das Schlüsselwort void in String verändert werden.
name darf in **printName()** erzeugt werden, ist dann jedoch nur innerhalb dieser Methode bekannt (*name* ist also eine Variable der Methode und kein Attribut der Klasse Robot).
Der Rückgabewert von **calculate()** ist int, also ganzzahlig. Damit die Berechnung der double-Werte *a-b*, also zweier Gleitkommazahlen, zurückgegeben werden kann, muss statt des Rückgabetyps int der Typ double genutzt werden.
Die Methode **printInfo()** erzeugt keinen Fehler. Java verkettet die Kombination eines Strings und eines int-Wertes *info + age* zu einem String, so dass diese Methode korrekt ist.
Falls du diese Frage nicht sicher beantworten konntest, schau dir am besten nochmal Video 1.7 Methoden und Rückgabewerte an.

- [x] Die Methode **calculate()** wird einen Fehler erzeugen. Ihr Rückgabewert ist int, sie gibt aber die Berechnung der double-Werte *a-b* zurück. — Correct!
- [ ] die Methode **printInfo()** wird einen Fehler erzeugen. Ihr Rückgabewert ist ein String, sie gibt aber die Kombination eines Strings und eines int-Werts *info + age* zurück.
- [ ] Das Codebeispiel lässt sich fehlerfrei ausführen.
- [ ] *name* darf nicht innerhalb von **printName()** angelegt werden.
- [x] Die Methode **printName()** wird einen Fehler erzeugen. Ihr Rückgabewert ist void, sie gibt aber den in *name* gespeicherten String zurück. — Correct!

97%

Total: **19.5 of 20.0 points achieved**

This quiz can be repeated indefinitely!

Your submissions: 2

Choose submission:

Wed, Aug 23, 2023 16:18:08