Quizzies

Aktuelle Punktzahl: 0 / 19

Beantwortet: 0 / 20

`Roesti` ist eine Subklasse von `Speise` und folgende Zeile ist gegeben:

```
Speise essen = new Roesti();
```

Welche Aussage ist korrekt?

- [x] `Roesti` ist der dynamische und `Speise` der statische Typ
- [ ] `Roesti` ist der statische und `Speise` der dynamische Typ

➔ 1 Punkt

Quizzies

Aktuelle Punktzahl: 1 / 19

Beantwortet: 1 / 20

Welche Aussagen im Zusammenhang mit statischen und dynamischen Typen sind richtig?

- [x] Zwischen statischem und dynamischem Typ wird nur bei Datenfeldern unterschieden, bei lokalen Variablen sind die beiden Typen immer identisch
- [x] Der dynamische Typ einer Variablen muss derselbe Typ oder ein Subtyp (Subklasse) des statischen Typs sein
- [x] Statischer und dynamischer Typ einer Variablen werden beim Kompilieren festgelegt und beide sind während der Laufzeit nicht mehr veränderbar
- [ ] Der Compiler berücksichtigt den statischen Typ einer Variablen

➔ 0 Punkte

Quizzies

Test beenden Te

Aktuelle Punktzahl: 1 / 19

Beantwortet: 2 / 20

Welche Aussagen bez. Überschreiben von Methoden sind korrekt?

- [x] `B` ist Subklasse von `A`. Wenn `B` eine Methode von `A` überschreibt, dann muss dazu `A` nicht verändert werden
- [ ] Überschreiben und Überladen sind zwei unterschiedliche Bezeichnungen für dasselbe Konzept
- [ ] Die Klasse `A` hat zwei Subklassen `Sub1` und `Sub2` und implementiert die Methode `zeige()`. Wenn `Sub1` diese Methode überschreibt, dann muss sie auch in `Sub2` überschrieben werden.
- [x] Um eine Methode zu überschreiben, muss dieselbe Signatur wie bei der überschriebenen Methode verwendet werden

➔ 1 Punkt

Quizzies

Test beenden

Aktuelle Punktzahl: 2 / 19

Beantwortet: 3 / 20

Ein Programm definiert eine Klasse `Planet`, die keine explizite Superklasse aber diverse Subklassen hat. Was muss gelten, damit bei der letzten Zeile kein Kompilierfehler erzeugt wird?

```
Planet planet;
...
planet.zeige();
```

- [x] Die Klasse `Planet` muss die Methode `zeige()` direkt implementieren und alle Subklassen müssen sie überschreiben
- [ ] Die Klasse `Planet` muss die Methode `zeige()` direkt implementieren und das momentan von `planet` referenzierte Objekt muss die Methode überschreiben
- [x] Die Klasse `Planet` muss die Methode `zeige()` implementieren

^

➔ 0 Punkte

Quizzies

Test

Aktuelle Punktzahl: 2 / 19

Beantwortet: 4 / 20

`B` sei Subklasse von `A` und `C` sei Subklasse von `B`. Wenn `A` eine Methode implementiert und `B` diese Methode nicht überschreibt, darf sie dann von `C` überschrieben werden?

- [ ] Nein
- [x] Ja

➔ 1 Punkt

Quizzies

Test beenden | Test unterbr

Aktuelle Punktzahl: 3 / 19

Beantwortet: 5 / 20

Welche Aussagen zur Methodensuche sind korrekt?

- [x] Es wird immer die Methode des dynamischen Typs verwendet. Ist die Methode in der entsprechenden Klasse nicht direkt implementiert, so gibt es einen Laufzeitfehler
- [x] Zur Kompilierzeit ist im Allgemeinen nicht bekannt, welche Methode zur Laufzeit aufgerufen wird
- [ ] Es wird immer die Methode des statischen Typs der Variable verwendet
- [x] Es wird immer die Methode des dynamischen Typs verwendet. Diese Methode kann direkt in der entsprechenden Klasse implementiert bzw. überschrieben oder von einer Superklasse geerbt sein.

➔ 0 Punkte

Quizzies

Tes

Aktuelle Punktzahl: 3 / 19

Beantwortet: 6 / 20

Welche Aussagen bez. der Verwendung von `super` in überschriebenen Methoden sind korrekt?

- [x] Die Verwendung ist fakultativ: Im Gegensatz zu Konstruktoren muss `super` also nicht zwingend verwendet werden
- [x] Die Anweisung muss an erster Stelle stehen, genau wie bei den Konstruktoren
- [ ] Wird `super` nicht verwendet, so fügt der Compiler wie bei Konstruktoren zu Beginn der Methode einen super-Aufruf der überschriebenen Mehode ohne Argumente ein: `super.ueberschriebeneMethode();`
- [x] `super` in der überschreibenden Methode ruft die überschriebene Methode auf

➔ 0 Punkte

Quizzies

Test beenden | Test unterbrechen

Aktuelle Punktzahl: 3 / 19

Beantwortet: 7 / 20

`B` sei Subklasse von `A` und `C` sei Subklasse von `B`. `A` implementiert eine Methode, welche in `B` nicht überschrieben wird. `C` überschreibt die Methode. Darf dann `super` in der überschriebenen Methode von `C` überhaupt verwendet werden?

- [ ] Nein
- [x] Ja

➔ 1 Punkt

Quizzies

Test beenden | Test un

Aktuelle Punktzahl: 4 / 19

Beantwortet: 8 / 20

Die Klasse `Hund` erweitert die Klasse `Haustier` und die Klasse `Dogge` die Klasse `Hund`. Sowohl `Hund` als auch `Dogge` implementieren die Methode `public void bellen() {...}`. Welche `bellen`-Methode wird bei der letzten Programmzeile ausgeführt?

```
Hund hund = new Dogge();
hund.bellen();
```

- [x] Die von `Dogge`
- [ ] Es gibt einen Kompilierfehler
- [ ] Es gibt einen Laufzefehler
- [x] Die von `Hund`

➔ 0 Punkte

Quizzies

Test beenden | Test ur

Aktuelle Punktzahl: 4 / 19

Beantwortet: 9 / 20

Die Klasse `Hund` erweitert die Klasse `Haustier` und die Klasse `Dogge` die Klasse `Hund`. Sowohl `Hund` als auch `Dogge` implementieren die Methode `public void bellen() {...}`. Welche `bellen`-Methode wird bei der letzten Programmzeile ausgeführt?

```
Haustier tier = new Dogge();
tier.bellen();
```

- [ ] Die von `Hund`
- [ ] Es gibt einen Laufzefehler
- [x] Die von `Dogge`
- [ ] Es gibt einen Kompilierfehler

Tier = statischer Typ
Dogge = dynamischer Typ

➔ 0 Punkte

Quizzies

Test beenden | Test ur

Aktuelle Punktzahl: 4 / 19

Beantwortet: 10 / 20

Die Klasse `Hund` erweitert die Klasse `Haustier` und die Klasse `Dogge` die Klasse `Hund`. Sowohl `Hund` als auch `Dogge` implementieren die Methode `public void bellen() {...}`. Welche `bellen`-Methode wird bei der letzten Programmzeile ausgeführt?

```
Haustier tier = new Hund();
((Dogge) tier).bellen();
```

- [ ] Es gibt einen Kompilierfehler
- [x] Die von `Dogge`
- [ ] Die von `Hund`
- [ ] Es gibt einen Laufzeitfehler

➔ 0 Punkte

Quizzies

Test beenden | Test ur

Aktuelle Punktzahl: 4 / 19

Beantwortet: 11 / 20

Die Klasse `Hund` erweitert die Klasse `Haustier` und die Klasse `Dogge` die Klasse `Hund`. Sowohl `Hund` als auch `Dogge` implementieren die Methode `public void bellen() {...}`. Welche `bellen`-Methode wird bei der letzten Programmzeile ausgeführt?

```
Haustier tier = new Dogge();
((Dogge) tier).bellen();
```

- [ ] Es gibt einen Laufzefehler
- [ ] Es gibt einen Kompilierfehler
- [ ] Die von `Hund`
- [x] Die von `Dogge`

➔ 1 Punkt

Quizzies

Aktuelle Punktzahl: 5 / 19

Beantwortet: 12 / 20

Sie können von jedem beliebigen Objekt, das Sie in einem Java Programm erzeugen, immer die `toString()`- und `equals()`-Methoden aufrufen. Stimmt das?

- [x] Ja
- [ ] Nein

➔ 1 Punkt

Quizzies

Test be

Aktuelle Punktzahl: 6 / 19

Beantwortet: 13 / 20

Sie überschreiben in einer Klasse `Pudel` die `toString()`-Methode. `pudel` sei ein Objekt vom Typ `Pudel`. Was ist der Unterschied zwischen den beiden folgenden Aufrufen?

```
System.out.println(pudel.toString());
System.out.println(pudel);
```

- [ ] Die zweite Variante ergibt einen Kompilierfehler
- [x] Keiner, in beiden Fällen wird der Rückgabewert der überschriebenen toString-Methode ausgegeben
- [ ] Beides funktioniert, aber im ersten Fall wird die überschriebene `toString()`-Methode und im zweiten Fall die `toString()`-Methode der Klasse `Object` aufgerufen

➔ 1 Punkt

Quizzies

Test beenden | Test unterbrechen

Aktuelle Punktzahl: 7 / 19

Beantwortet: 14 / 20

Die Klasse `Frosch` hat keine explizite Superklasse und überschreibt die `equals()`-Methode nicht. Sie erzeugen zwei `Frosch`-Objekte, die inhaltlich völlig identisch sind (gleiche Werte für die Datenfelder) und referenzieren sie mit den Variablen `quaki` und `froggy`, die beide vom Typ `Frosch` sind.
Betrachten Sie nun die folgende Bedingung. Welche Antworten sind korrekt?

```
if (quaki.equals(froggy)) ...
```

- ☐ Da die Datenfelder beider Objekte identisch sind, ist die Bedingung true.
- ☒ Für einen inhaltlichen Vergleich muss `equals()` überschrieben werden
- ☒ Die Bedingung ist identisch zu `if (quaki == froggy) ...`
- ☑ Die Bedingung ist false, ausser beide Variablen referenzieren dasselbe Objekt

➔ 0 Punkte

Quizzies

Test beenden | Test untert

Aktuelle Punktzahl: 7 / 19

Beantwortet: 15 / 20

Wenn man bei der Entwicklung der Klasse `A` bereits weiss, dass `A` durch Subklassen erweitert wird, dann entspricht es den Clean Code-Regeln, wenn man alle Datenfelder von `A` `protected` macht.

- ☐ Ja
- ☒ Nein

➔ 0 Punkte

Quizzies

Test beenden | Test unterbrechen

Aktuelle Punktzahl: 7 / 19

Beantwortet: 16 / 20

`B` ist Subklasse von `A` und `C` ist Subklasse von `B`. `A` hat ein privates Datenfeld `hoehe` und eine `protected` Methode `gibHoehe()`, welche den Wert des Datenfelds zurückliefert. Welche Aussagen sind korrekt?

- ☑ In einem Objekt der Klasse `B` kann die Methode verwendet werden und es wird der korrekte Wert geliefert
- ☐ In einem Objekt der Klasse `A` kann die Methode nicht verwendet werden, da `protected` Methoden immer nur in Subklassen verwendet werden dürfen
- ☑ In einem Objekt der Klasse `C` kann die Methode verwendet werden und es wird der korrekte Wert geliefert
- ☐ In einem Objekt der Klasse `B` kann die Methode zwar verwendet werden, sie liefert aber 0 statt des korrekten Werts, da der Zugriff auf private Datenfelder der Superklasse aus der Subklasse grundsätzlich nicht möglich ist

➔ 1 Punkt

Quizzies

Aktuelle Punktzahl: 8 / 19

Beantwortet: 17 / 20

`Hund` ist Subklasse von `Haustier` und `Dogge` von `Hund`. Was liefert die Bedingung in der zweiten Zeile für einen Wert?

```
Haustier tier = new Dogge();
if (tier instanceof Dogge) ...
```

- ☑ true
- ☐ false

➔ 1 Punkt

Quizzies

Aktuelle Punktzahl: 9 / 19

Beantwortet: 18 / 20

`Hund` ist Subklasse von `Haustier` und `Dogge` von `Hund`. Was liefert die Bedingung in der zweiten Zeile für einen Wert?

```
Haustier tier = new Dogge();
if (tier instanceof Hund) ...
```

- [x] true
- [ ] false

➔ 1 Punkt