JAVA 02: Bool'sche Ausdrücke und Verzweigungen.

panitz

Zusammenfassung

25 Fragen über Ausdrücke und Anweisungen mit bool'schen Werten. In diesem Kurs befinden sich 25 einfache Fragen, die sich um Vergleiche, bool'sche Werte und Verzweigungen drehen.

Frage: Vergleichsoperatoren (0)

Der »kleiner-gleich«-Vergleichsoperator ist?

(Eine Antwortmöglichkeit.)

- <
- <=
- =<
- >=
- "=

Korrekte Antworten

- <=

Erläuterung

Die Operatoren =< und "= gibt es nicht. < ist der Operator für echt kleiner.

Frage: Vergleichsoperatoren (1)

Der Gleichheits-Vergleichsoperator ist?

(Eine Antwortmöglichkeit.)

- `<>`
- `!=`
- `==`
- `=`

Korrekte Antworten

- ==

Erläuterung

Der Gleicheitsoperator besteht aus zwei Gleichheitszeichen: ==. Ein Gleichheitszeichen = steht für eine Zuweisung.

Frage: Programmanalyse (0)

Es sei `boolean isPrime` eine Variable. Welches der folgenden Teilstatements ist in Java der korrekte Beginn eine Anweisung, um zu verzweigen, wenn `isPrime` wahr ist?

(Mehrere Antwortmöglichkeiten).

- `if (isPrime = true)`
- `if (isPrime == true)`
- `if (isPrime)`
- `if (!isPrime == false)`
- `if (isPrime == false)`

Korrekte Antworten

- `if (isPrime == true)`
- `if (isPrime)`
- `if (!isPrime == false)`

Erläuterung

Der Ausdruck `(isPrime = true)` weist `isPrime` den Wert `true` zu und wertet somit immer zu `true` aus. Auf `isPrime == true` sollte man verzichten, denn `isPrime` ist bereits der gewünschte Wahrheitswert. `(!isPrime == false)` ist unnötig komplex.

Frage: Programmanalyse (1)

Was druckt das folgende Codestückchen auf der Kommandozeile aus?

```
int x = ;
if (x<) x = x + ;
System.out.println("x ist "+x);
```

(Eine Antwortmöglichkeit.)

- `x ist`
- `x ist`
- `x ist`
- `x ist`
- `x ist`

Korrekte Antworten

- `x ist`

Erläuterung

Die Bedingung der if-Anweisung wertet zu `true` aus. Der Rumpf der if-Anweisung wird entsprechend ausgeführt und x um 3 erhöht.

Frage: Programmanalyse (2)

Analysieren Sie den folgenden Code.

```
boolean even = false;
if (even = true)
  System.out.println("It is even!");
```

(Eine Antwortmöglichkeit.)

- Das Programm erzeugt einen Syntax-Error wegen Zeile 1: `boolean even = false;`
- Das Programm erzeugt einen Syntax-Error wegen Zeile 2: `if (even = true)`
- Das Programm läuft, druckt aber nichts aus.
- Das Programm läuft und druckt `It is even!` aus.

Korrekte Antworten

- Das Programm läuft und druckt `It is even!` aus.

Erläuterung

Böse kleine Falle: `(even = true)` weist der Variablen `even` den Wert `true` zu und wertet entsprechend zu `true` aus.

Frage: Programmanalyse (3)

Was druckt das folgende Programm aus?

```
int x = -1;
int y = -1;
int z = 1;

if (x>0){
  if (y>0) {
    System.out.println("x > 0 and y > 0");
  }
}else if (z>0){
  System.out.println("x < 0 and z > 0");
}
```

(Eine Antwortmöglichkeit.)

- `x > 0 and y > 0`
- `x < 0 and z > 0`
- `x < 0 and z < 0`
- Es wird nichts ausgedruckt.

Korrekte Antworten

- `x < 0 and z > 0`

Erläuterung

Da x initial nicht größer 0 ist, wird in den else-Fall der äußeren Verzweigung gesprungen. Die Bedingung der if-Verzweigung dort wertet zu `true` aus.

Frage: Programmanalyse (4)

Der folgende Code druckt was aus?

```
int temperature = ;
if (temperature >= )
  System.out.println("zu heiß");
else if (temperature <= )
  System.out.println("zu kalt");
else
  System.out.println("gut");
```

(Eine Antwortmöglichkeit.)

- zu heiß
- zu kalt
- gut
- zu heiß
  zu kalt
  genau richtig

Korrekte Antworten

- `gut`

Erläuterung

Es wird jeweils in den else-Zweig der if-Verzweigungen gesprungen.

Frage: Programmanalyse (5)

Betrachten Sie die beiden Programme:
Programm 1:

```
public static void main(String[] args){
  int number = new Integer(args[0]);
  boolean even = false;
  if (number%2==0) even = true;
  else even = false;}
```

Programm 2:

```
public static void main(String[] args){
  int number = new Integer(args[0]);
  boolean even = number%2==0; }
```

Welche Aussagen sind korrekt?

(Mehrere Antwortmöglichkeiten).

- Programm 1 endet immer mit einem Fehler.
- Programm 2 endet immer mit einem Fehler.
- Beide Programme laufen fehlerfrei bei Übergabe einer Zahl auf der Kommandozeile, aber Programm 1 ist schneller.
- Beide Programme haben beim Laufen in allen Aspekten das gleiche Verhalten.

Korrekte Antworten

- Beide Programme haben beim Laufen in allen Aspekten das gleiche Verhalten.

Erläuterung

Wie man sieht ist die zweite Version vorzuziehen, weil direkt ein bool'scher Ausdruck berechnet und der Variablen `even` zugewiesen wird.

Frage: Programmanalyse (6)

Welche Ausgabe druckt nachstehendes Programm auf der Kommandozeile aus?

```
int income = ;
if (income > )
  System.out.println("Einkommen ist größer als .");
else if (income > )
  System.out.println("Einkommen ist größer als .");
```

(Eine Antwortmöglichkeit.)

- `Einkommen ist größer als .`
- `Einkommen ist größer als .`
  `Einkommen ist größer als .`
- `Einkommen ist größer als .Einkommen ist größer als .`
- `Einkommen ist größer als .`
- `Einkommen ist größer als .`
  `Einkommen ist größer als .`

Korrekte Antworten

- `Einkommen ist größer als .`

Erläuterung

Da die Bedingung der Verzweigung wahr ist, wird nicht mehr in den else-Fall gesprungen.

Frage: Programmanalyse (7)

Sie wollen ein Programm schreiben, das »Nein« (ohne die Anführungszeichen) ausdruckt, wenn das Alter (age) < 18 ist. Aber »Ja« (ohne die Anführungszeichen) soll ausgedruckt werden, sofern das Alter (age) größer oder gleich 18 ist. Analysieren Sie die folgenden fünf Alternativen und markieren Sie diejenigen, die das Gewünschte ausgeben.

(Mehrere Antwortmöglichkeiten).

- ```
  if (age < 18) System.out.println("Nein");if (age >= 18) System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");else System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else if (age >= 18) System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else System.out.println("Ja");
  if (age == 18)  System.out.println("Ja");
  else if (age>18) System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else if (age == 18) System.out.println("Ja");
  ```
  ```

Korrekte Antworten

- ```
  if (age < 18) System.out.println("Nein");if (age >= 18) System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");else System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else if (age >= 18) System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else if (age == 18) System.out.println("Ja");
  else if (age > 18) System.out.println("Ja");
  else System.out.println("Ja");
  ```

Erläuterung

Man muss jeweils dem Programmfluss bei der Analyse folgen.

Frage: Programmanalyse (8)

Markieren Sie die beste Lösung des Programms, das "Nein"(ohne die Anführungszeichen) ausdruckt, wenn das Alter (age) < 18 ist, aber "Ja"(ohne die Anführungszeichen) sofern das Alter (age) größer oder gleich 18 ist.
Analysieren Sie die folgenden vier Alternativen.

(Eine Antwortmöglichkeit.)

- ```
  if (age < 18) System.out.println("Nein");
  if (age >= 18) System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else System.out.println("Ja");
  ```

- ```
  if (age < 18) System.out.println("Nein");
  else if (age >= 18) System.out.println("Ja");
  ```

- ```
  int age = 19;
  if (age < 18) System.out.println("Nein");
  else if (age == 18) System.out.println("Ja");
  else if (age > 18) System.out.println("Ja");
  else System.out.println("Ja");
  ```

Korrekte Antworten

```
▪ if (age < 18) System.out.println("Nein");
  else System.out.println("Ja");
```

Erläuterung

Vorzuziehen ist natürlich die Version, die mit einer if-Verzweigung mit einem else-Fall auskommt.

Frage: Programmanalyse (9)

Was wird durch folgendes Codestückchen ausgedruckt?

```
char ch = 'F';
if (ch >= 'A' && ch <= 'Z')
  System.out.println(ch);
```

(Eine Antwortmöglichkeit.)

- `F`
- `f`
- nichts wird ausgedruckt
- `Ff`

Korrekte Antworten

- `F`

Erläuterung

Auch auf char-Werten lässt sich wie auf anderen Zahlenwerten rechnen und Vergleichsoperatoren anwenden.

Frage: Boolesche Operatoren (0)

Was ist das Wort `true` ?

(Mehrere Antwortmöglichkeiten).

- Ein Java Schlüsselwort.
- Ein boolsches Literal.
- Dasselbe wie der Wert 1.
- Dasselbe wie der Wert 0.

Korrekte Antworten

- Ein Java Schlüsselwort.
- Ein boolsches Literal.

Erläuterung

Wahrheitswerte sind in Java keine Zahlen und Zahlen können nicht als Wahrheitswerte verwendet werden.

Frage: Boolesche Operatoren (1)

Welche der folgenden Ausdrücke ergeben einen Fehler bei der Kompilierung?

(Mehrere Antwortmöglichkeiten).

- `true && 3 >= 4`
- `!(x > 0) && (x > 0)`
- `(x > 0) || (x < 0)`
- `(x != 0) || (x = 0)`
- `(-10 < x < 0)`

Korrekte Antworten

- `(x != 0) || (x = 0)`
- `(-10 < x < 0)`

Erläuterung

`x = 0` ist kein Wahrheitswert. Der Ausdruck `(-10 < x < 0)` soll wohl `(-10 < x && x < 0)` sein.

Frage: Boolesche Operatoren (2)

Was kann man schreiben, um zu überprüfen, ob der Wert der Variablen `ch` einen Großbuchstaben enthält?

(Eine Antwortmöglichkeit.)

- `(ch >= 'A' && ch >= 'Z')`
- `(ch >= 'A' && ch <= 'Z')`
- `(ch >= 'A' || ch <= 'Z')`
- `('A' <= ch <= 'Z')`

Korrekte Antworten

- `(ch >= 'A' && ch <= 'Z')`

Erläuterung

Wie man sieht, lässt sich auf char-Werten eine Vergleichsoperation anwenden.

Frage: Boolesche Operatoren (3)

Es sei $|x - 2| \leq$.
Welcher der folgenden Ausdrücke wertet sich (immer) zu `true` aus?

(Mehrere Antwortmöglichkeiten).

- `x - 2 <=  && x - 2 >=`
- `x - 2 <=  && x - 2 > -`
- `x - 2 <=  && x - 2 >= -`
- `x - 2 <=  || x - 2 >= -`

Korrekte Antworten

- `x - 2 <= && x - 2 >= -`
- `x - 2 <= || x - 2 >= -`

Erläuterung

x liegt im Intervall zwischen -4 und 8.

Frage: Boolesche Operatoren (3)

Es sei $|x - 2| \geq 4$.
Welcher der folgenden Ausdrücke wertet sich zu `true` aus?

(Eine Antwortmöglichkeit.)

- `x - 2 >= 4 && x - 2 < -4`
- `x - 2 > 4 || x - 2 <= -4`
- `x - 2 >= 4 && x - 2 <= -4`
- `x - 2 >= 4 || x - 2 <= -4`

Korrekte Antworten

- `x - 2 >= 4 || x - 2 <= -4`

Erläuterung

x liegt außerhalb des Intervalls von -2 bis 6.

Frage: Boolesche Operatoren (5)

Es sei $x =$ und $y =$.
Welcher der folgenden Ausdrücke wertet sich zu `true` aus?

(Eine Antwortmöglichkeit.)

- `x <  && y <`
- `x <  || y <`
- `(x > ) && (y > )`
- `(x > ) || (y > )`

Korrekte Antworten

- `x <  || y <`

Erläuterung

Da x kleiner ist, wird der Oder-Ausdruck wahr.

Frage: Boolesche Operatoren (6)

Es sei $x = 1$ und $y =$.
Welcher der folgenden Ausdrücke wertet sich zu `true` aus?

(Mehrere Antwortmöglichkeiten).

- `x % 2 == 0 && y % 2 == 0`
- `x % 2 == 0 && y % 2 == 1`
- `x % 2 == 0 || y % 2 == 0`
- `x % 2 != 0 && y % 2 != 0`

Korrekte Antworten

- `x % 2 == 0 && y % 2 == 1`
- `x % 2 == 0 || y % 2 == 0`

Erläuterung

x ist gerade und y ungerade.

Frage: Boolesche Operatoren (7)

Welcher der folgenden Ausdrücke ist äquivalent zu `x != y` ?

(Mehrere Antwortmöglichkeiten).

- `! (x == y)`
- `x > y && x < y`
- `x > y || x < y`
- `x >= y || x <= y`

Korrekte Antworten

- `! (x == y)`
- `x > y || x < y`

Erläuterung

Der Operator der Ungleichheit `!=` negiert das Ergebnis der Gleichheit.

Frage: Boolesche Operatoren (8)

Vergleichen Sie jeweils die Ausdrücke A) und B). Kreuzen Sie die Alternative an, bei denen A) und B) für alle x, y zum selben Wert ausgewertet wird.

(Eine Antwortmöglichkeit.)

- – A) `(x > 0 && x < 10)`
  – B) `(x > 0 && x > 10)`
- – A) `(x > 0 || x < 10)`
  – B) `(0 < x && x < 10)`
- – A) `(x > 0 || x < 10 && y < 0)`
  – B) `(x > 0 || (x < 10 && y < 0))`
- – A) `(x > 0 || x < 10 && y < 0)`
  – B) `((x > 0 || x < 10) && y < 0)`

Korrekte Antworten

- – A) `(x > 0 || x < 10 && y < 0)`
  – B) `(x > 0 || (x < 10 && y < 0))`

Erläuterung

Man beachte dass `&&` stärker bindet als `||`.

Frage: Conditional Expression (1)

Welchen Wert hat y nach der Ausführung des folgenden Code-Segments:

```
int x = 0;
int y = x > 0? 10 : -10;
```

(Eine Antwortmöglichkeit.)

- −10
- 0
- 10
- Das Programm kompiliert nicht.

Korrekte Antworten

- `-10`

Erläuterung

Der ternäre Operator `? :` macht eine Auswahl nach dem Wert des ersten Operanden.

Frage: Conditional Expression (2)

Analysieren Sie die drei folgenden Code-Fragmente, die einer Variablen `even` einen booleschen Wert zuweisen.
Welche korrekte Variante wird von den meisten Java-Programmierern bevorzugt?

- **Code 1:**
  `if (number % 2 == 0) even = true;else even = false;`
- **Code 2:**
  `even = number % 2 == 0 ? true : false`
- **Code 3:**
  `even = number % 2 == 0`

(Eine Antwortmöglichkeit.)

- Code 2 erzeugt einen SyntaxError, weil man in einem Conditional-Ausdruck nicht True und False gleichzeitig haben kann.
- Code 3 erzeugt einen SyntaxError, weil man versucht der Variablen even eine Zahl zuzuweisen.
- Alle drei Codes (Code1 bis Code3) sind korrekt, bevorzugt wird aber Code 1 genutzt.
- Alle drei Codes (Code1 bis Code3) sind korrekt, bevorzugt wird aber Code 2 genutzt.
- Alle drei Codes (Code1 bis Code3) sind korrekt, bevorzugt wird aber Code 3 genutzt.

Korrekte Antworten

- Alle drei Codes (Code1 bis Code3) sind korrekt, bevorzugt wird aber Code 3 genutzt.

Erläuterung

Das Ergebnis des Ausdrucks `number % 2 == 0` ist bereits der gesuchte Wahrheitswert. Eine Fallunterscheidung per `? :` Ausdruck oder if-Anweisung ist unnötig.

Frage: Conditional Expression (3)

Was wird von folgendem Code ausgedruckt?

```
boolean isCorrect = false;
System.out.println(isCorrect? "korrekt" : "falsch");
```

(Eine Antwortmöglichkeit.)

- `korrekt`
- `falsch`
- Es kommt zu keiner Ausgabe.
- `korrekt falsch`

Korrekte Antworten

- `falsch`

Erläuterung

Der Bedingungsoperator selektiert in diesem Fall den dritten Operanden.

Frage: Boolesche Ausdrücke

Zu was wertet folgender Ausdruck aus:
`(x=!false) && (x!=!true)`

(Eine Antwortmöglichkeit.)

- Das lässt sich nich sagen, wenn man den initialen Wert von `x` nicht kennt.
- false, denn `y` kann nicht gleichzeitig ungleich false und ungleich true sein.
- Dieser Ausdruck wird vom Compiler zurückgewiesen.
- true wegen der Klammerung.

Korrekte Antworten

- true wegen der Klammerung.

Erläuterung

Ja, das ist gemein. Der erste Operand der logischen Konjunktion ist eine Zuweisung. Danach hat `x` den Wert true. Und damit ist der Ausdruck `x!=!true` auch true. Ohne die Klammern `x=!false && x!=!true` wäre der Ausdruck wie folgt geklammert: `x = (!false && x!=!true)`. Damit hinge es vom ursprüngliehcne Wert von `x` ab. Die Moral von der Geschichte: Ausdrücke und Zuweisungen besser nicht mischen.