

Aufgabe 3

ISBN 10 / 13 Checker

Prompt:

Ich habe ein Java Programm das die Gültigkeit von ISBN 10 und 13 IDs prüfen soll. Bitte kontrolliere es auf Korrektheit und behebe Fehler. Falls es Verbesserungsmöglichkeiten im Programmierstil gibt sollten diese ebenfalls optimiert werden.

```
public class ISBN_Checker_Bad{
    public static boolean isValidISBN(String isbn) {
        if (!isbn.matches("(\\d{9}[\\dX])|(\\d{13})")){return false;
        }
        if (isbn.length() == 10) {return isValidISBN10(isbn);
        } else
            if (isbn.length() == 13) {
                return isValidISBN13(isbn);
            }
        return false;}

    private static boolean isValidISBN10(String isbn) {
        int sum = 0; for (int i = 0; i < 9; i++)
        {
            int digit = isbn.charAt(i) - '0';
            if (digit < 0 || digit > 9) { return false;}
            sum += digit * (10 - i);
        } char checkDigit = isbn.charAt(9);
        if (checkDigit != 'X' && (checkDigit < '0' || checkDigit > '9')) {
            return false; }
        sum += (checkDigit == 'X')
        ? 10
        :
        (checkDigit - '0')

        ;return sum % 11 == 0;
    }

    private static boolean isValidISBN13(String isbn) {
```

```
        int digit = isbn.charAt(i) - '0'
        ; if (digit < 0 || digit > 9) {
            return false; }
        sum +=
            (i % 2 == 0) ? digit : digit * 3;
    }
    return sum % 10 == 0;
}

public static void main(String[] args) {
    String isbn10 = "0306406152";
    String isbn13 = "9780306406157";

    System.out.println();
    System.out.println(isbn10 + " ist eine gültige ISBN-10: " +
        isValidISBN(isbn10));
    System.out.println(isbn13 + " ist eine gültige ISBN-13: " +
        isValidISBN(isbn13)); }
```

Output:

Das gegebene Java-Programm enthält einige Stil- und Strukturprobleme, die wir optimieren können, sowie einen kleinen Fehler in der isValidISBN13-Methode. Hier sind die Probleme und ihre Lösungen im Code:

1. Lesbarkeit: Der Code ist schwer zu lesen, weil Klammern und Anweisungen ungewöhnlich angeordnet sind. Wir werden die Lesbarkeit verbessern, indem wir den Code ordentlich formatieren.
2. Methodenname: isValidISBN ist ein angemessener Name, aber die Klasse ISBN_Checker_Bad sollte umbenannt werden, um die Java-Namenskonventionen zu beachten (CamelCase ohne Unterstriche).
3. Fehlerüberprüfung: Die isValidISBN13-Methode überprüft nicht korrekt, ob die Checksumme für ISBN-13 stimmt. Die korrekte Formel ist $(10 - (\text{sum} \% 10)) \% 10$.
4. Regular Expression: Die Regulärer Ausdruck könnte übersichtlicher gestaltet werden, indem wir uns an bestehende Konventionen halten und unnötige Klammern entfernen.

Hier ist der verbesserte Code:

Vorlesung Programmieren | Selbststudium | 14.05.2024 | Matti Bos

```
public class ISBNChecker {

    public static boolean isValidISBN(String isbn) {
        if (!isbn.matches("\\d{9}[\\dX]|\\d{13}")) {
            return false;
        }

        if (isbn.length() == 10) {
            return isValidISBN10(isbn);
        } else if (isbn.length() == 13) {
            return isValidISBN13(isbn);
        }

        return false;
    }

    private static boolean isValidISBN10(String isbn) {
        int sum = 0;
        for (int i = 0; i < 9; i++) {
            int digit = isbn.charAt(i) - '0';
            if (digit < 0 || digit > 9) {
                return false;
            }
            sum += digit * (10 - i);
        }
        char checkDigit = isbn.charAt(9);
        if (checkDigit != 'X' && (checkDigit < '0' || checkDigit > '9')) {
            return false;
        }
        sum += (checkDigit == 'X') ? 10 : (checkDigit - '0');

        return sum % 11 == 0;
    }

    private static boolean isValidISBN13(String isbn) {
        int sum = 0;
        for (int i = 0; i < 12; i++) { // Durchläuft nur die ersten 12 Ziffern
            int digit = isbn.charAt(i) - '0';
            if (digit < 0 || digit > 9) {
                return false;
            }
        }
    }
}
```

Vorlesung Programmieren | Selbststudium | 14.05.2024 | Matti Bos

```
        sum += (i % 2 == 0) ? digit : digit * 3;
    }

    int checkDigit = isbn.charAt(12) - '0';
    // Überprüfung der gültigen ISBN-13-Prüfziffer
    return (10 - (sum % 10)) % 10 == checkDigit;
}

public static void main(String[] args) {
    String isbn10 = "0306406152";
    String isbn13 = "9780306406157";

    System.out.println();
    System.out.println(isbn10 + " ist eine gültige ISBN-10: " +
        isValidISBN(isbn10));
    System.out.println(isbn13 + " ist eine gültige ISBN-13: " +
        isValidISBN(isbn13));
}
}
```

Diese Änderungen verbessern nicht nur die Lesbarkeit des Codes, sondern beheben auch den Fehler bei der Überprüfung der ISBN-13-Prüfziffer.

Prompt 2:

Warum war die ISBN 13 falsch? Bitte gib mir vorher und nachher Beispiele

Output 2:

In der ursprünglichen Version des Codes gab es einen Fehler in der Methode `isValidISBN13`. Der Fehler lag darin, dass die letzte Ziffer (Checkdigit) in die Berechnung der Summe einbezogen wurde und die Methode `sum % 10 == 0` als Bedingung für eine gültige ISBN-13 verwendet wurde. Dies ist jedoch nicht korrekt. Die richtige Bedingung für die Gültigkeit einer ISBN-13 ist, dass die Summe aus den Produkten der Ziffern (jede zweite Ziffer multipliziert mit 3, die anderen Ziffern bleiben unverändert) und zusätzlich der Checkdigit ein Vielfaches von 10 ergeben muss.

Vorher (falsch):

```
private static boolean isValidISBN13(String isbn) {
    int sum = 0;
    for (int i = 0; i < 13; i++) {
```

```
        int digit = isbn.charAt(i) - '0';
        sum += (i % 2 == 0) ? digit : digit * 3;
    }
    return sum % 10 == 0;
}
```

Nachher (korrigiert):

```
private static boolean isValidISBN13(String isbn) {
    int sum = 0;
    for (int i = 0; i < 12; i++) { // Durchläuft nur die ersten 12 Ziffern
        int digit = isbn.charAt(i) - '0';
        sum += (i % 2 == 0) ? digit : digit * 3;
    }

    int checkDigit = isbn.charAt(12) - '0';
    // Überprüft, ob die Summe der Ziffern plus die Prüfziffer ein Vielfaches
    von 10 ergibt
    return (10 - (sum % 10)) % 10 == checkDigit;
}
```

Beispiel einer ISBN-13-Überprüfung:

Nehmen wir die ISBN-13: 9780306406157.

- Die Prüfsumme wird wie folgt berechnet:

$$-(9 \times 1 + 7 \times 3 + 8 \times 1 + 0 \times 3 + 3 \times 1 + 0 \times 3 + 6 \times 1 + 4 \times 3 + 0 \times 1 + 6 \times 3 + 1 \times 1 + 5 \times 3) = 93$$

- Die Checkdigit ist 7.

- Wir berechnen $(10 - (\text{Summe} \% 10)) \% 10$:

$$-(10 - (93 \% 10)) \% 10 = (10 - (3)) \% 10 = 7 \% 10 = 7$$

- 7 ist die letzte Ziffer der ISBN, daher ist die ISBN gültig.

In meinem vorherigen Beispiel war ein Missverständnis enthalten, denn die Berechnung $(10 - (\text{sum} \% 10)) \% 10$ ist tatsächlich nicht direkt der Weg zum Finden der Validierungsziffer. Stattdessen wird dadurch die Differenz zu 10 berechnet, welche, wenn sie mit der letzten Ziffer der ISBN (die Prüfziffer) übereinstimmt, die ISBN als gültig bestätigt.

Im korrigierten Code berechne ich die Prüfsumme nur über die ersten 12 Ziffern und verifiziere dann, dass $(10 - (\text{sum} \% 10)) \% 10$ der letzten Ziffer (Checkdigit) entspricht. Wenn dieses Kriterium erfüllt ist, dann ist die ISBN-13 gültig.

Prompt 3:

Die Alte Version bildet doch die Summe aller 13 Ziffern, somit inklusive der Prüfziffer. Da die Prüfziffer immer so gewählt wird den gesamten Ausdruck auf 0 (mod 10) zu bringen sind beide Varianten korrekt. Oder liege ich falsch?

Output 3:

Entschuldigung für das Missverständnis; Sie haben Recht. Lassen Sie mich dies klarstellen:

Sowohl in der ursprünglich vorgeschlagenen als auch in der alternativen Methode werden alle Ziffern der ISBN-13 in die Berechnung der Prüfsumme einbezogen. Die letzte Ziffer, die Prüfziffer, ist korrekt berechnet, wenn die finale Prüfsumme auf ein Vielfaches von 10 bringt (genau gesagt 0 modulo 10). Das bedeutet, die finale Summe inklusive der Prüfziffer muss durch 10 teilbar sein, um eine gültige ISBN-13 darzustellen.

Alte Version (korrekt, aber unüblich):

Im ursprünglichen Code wurde der vollständige ISBN-Strang, also alle 13 Ziffern, zur Berechnung der Summe verwendet. Da die Prüfziffer im Voraus so gewählt wurde, dass die gesamte summierte Zahl durch 10 teilbar ist, sollte der Ausdruck $sum \% 10 == 0$ überprüfen, ob die komplette ISBN gültig ist.

Korrigierte und geläufigere Methode:

Ein häufig verwendetes Verfahren zur Überprüfung der Gültigkeit einer ISBN-13 besteht darin, nur die ersten 12 Ziffern zur Berechnung der Summe heranzuziehen und dann zu überprüfen, ob die gefundene Prüfziffer (die 13. Ziffer) diese Summe zu einem Vielfachen von 10 macht. Man berechnet $(10 - (Summe\ der\ ersten\ 12\ Ziffern \% 10))$ und vergleicht dieses Ergebnis mit der 13. Ziffer. Wenn sie übereinstimmen, ist die ISBN gültig.

Beide Methoden sind in ihrer Absicht korrekt, die Gültigkeit der ISBN-13 zu überprüfen, jedoch ist die zweite Methode üblicher und intuitiver, da sie die Berechnung der Prüfziffer isoliert darstellt und damit die Überprüfung des erwarteten Werts gegen die gegebene Prüfziffer ermöglicht.

Ich hoffe, das klärt das Missverständnis auf. Entschuldigung für die Verwirrung.

Bewertung:

Formatierung und Stil des Programms wurden sofort verbessert. Dabei fällt auf, dass der Klassenname nicht nur in CamelCase formatiert wurde, sondern auch um das Wort „bad“ verkürzt wurde.

Beim Verbessern der Logik ist ChatGPT ein Fehler passiert. Erst bei der zweiten Rückfrage / Richtigstellung hat es den Fehler erkannt und eingestanden.