Übungen: Inhalte in Methoden auslagern

Akzeptieren Sie das Github-Classroom-Assignment

Akzeptieren Sie das Github-Assignment unter folgendem Link und klonen Sie das Projekt in Ihre IDE:

2024/03/19 08:26

LINK in MOODLE einfügen

Aufgaben

0. Programm ausführen

Führen Sie das Programm `UserDialog.java` aus und machen Sie sich mit den Funktionen vertraut.

1. Menüauswahl in Methode auslagern

Wir haben uns vorgenommen, dass wir Duplikate im Code vermeiden möchten. Im Programm `UserDialog` listen wir dem User an zwei Stellen im Code seine Auswahlmöglichkeiten auf:

```
// Menue Anzeigen
System.out.println();
System.out.println("===============================================");
System.out.println("Was möchten Sie mit dieser Applikation machen?");
System.out.println("Wählen Sie 'r' für Taschenrechner");
System.out.println("Wählen Sie 'h' für Hasen und Hühner Problem");
System.out.println("Wählen Sie 's' für Soccermanager");
System.out.println("Wählen Sie 'x' um das Programm zu beenden!");
System.out.print(">> ");
```

Wir möchten diesen Programmteil nun in eine externe Methode packen. Wir arbeiten uns also durch die 3 Schritte durch:

1. Schnittstelle vereinbaren
2. Methodenkopf schreiben
3. Logik programmieren

Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: Menüauswahl in Methode auslagern

2. Unterprogramme auslagern

Um die Übersichtlichkeit des Codes zu erhöhen, lagern wir die **Unterprogramme (Rechner, Hasen- und Hühnerproblem und Soccermanager) in eigene Methoden** aus.

Um diese Unterprogramme nun in eine externe Methoden zu packen, arbeiten uns für jede Methode durch die 3 Schritte durch:

1. Schnittstelle vereinbaren
2. Methodenkopf schreiben
3. Logik programmieren

Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: Unterprogramme auslagern

3. Inputvalidierung auslagern

Wir haben noch duplizierten Code in unserer Anwendung.

Sowohl in der Methode `rechner()` wie auch in der Methode `hasenUndHuehner()` machen wir eine Inputvalidierung. Wir überprüfen mehrfach, ob ein `int` respektive ein `double` eigegeben wurde.

Wir können diese Funktionalität auslagern und so einfach wiederverwenden.

```
while(!scanner.hasNextDouble()){
    System.out.println("Bitte eine Zahl eingeben");
    scanner.nextLine();
}
zahl1 = scanner.nextDouble();
scanner.nextLine();
```

Um diese Validierung nun in eine externe Methoden zu packen, arbeiten uns für beide Methoden durch die 3 Schritte durch:

1. Schnittstelle vereinbaren
2. Methodenkopf schreiben
3. Logik programmieren

Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: Inputvalidierung auslagern

4. Code wiederverwenden

Wir haben nun eine einfache Methode geschrieben um sowohl `int` wie auch `double` mit Validierung einzulesen. In unserer Methode `soccermanager()` verwenden wir noch keine Inputvalidierung für das Einlesen von Zahlen.

Ergänzen Sie den Code der Methode `soccermanager()` und verwenden Sie überall wo Sie Werte einlesen die Methode `getValidInt()` oder `getValidDouble()`

Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: Code wiederverwenden

5. Letzte Optimierungen

Schauen wir uns verschiedene Codestellen nochmals genauer an:

```
System.out.print("Bewertung (1.0 - 6.0 / 0=Ende) >");
rating = getValidDouble();
```

```
System.out.print("Anzahl Beine ungültig, bitte erneut eingeben (Min:
"+minLegs +" Max: " + maxLegs +") >");
legs = getValidInt();
```

```
System.out.print("Anzahl Tiere >");
animals = getValidInt();
```

Zu jedem Einlesen einer Zahl gehört ein Dialog mit dem User. Wir fragen den User etwas und seine Antwort lesen wir dann ein.

Idee: Wir passen die Methoden `getValidInt()` und `getValidDouble()` an, das diese auch den `System.out.print` für uns übernehmen.

Arbeiten Sie sich für beide Methoden durch die 3 Schritte durch:

1. Schnittstelle vereinbaren
2. Methodenkopf schreiben
3. Logik programmieren

Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: Letzte Optimierungen

© Kevin Maurizi

2024/03/19 08:26

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m319/learningunits/lu07/aufgaben/methoden

Last update: **2024/03/28 14:07**