Unterrichtsmaterialien in digitaler und in gedruckter Form

Auszug aus:

Kontrollstrukturen in Java: Anwendung von zählergesteuerten Schleifen

Das komplette Material finden Sie hier:

School-Scout.de


Inhaltsfeld Algorithmen

Kontrollstrukturen in Java: Anwendung von zählergesteuerten Schleifen

Ein Beitrag von Johann-Georg Vogelhuber

© RAABE 2021

© Sitthiphong/iStock/Getty Images Plus

Zählergesteuerte Schleifen sind eine grundlegende Kontrollstruktur in jeder imperativen Programmiersprache. Lassen Sie Ihre Schülerinnen und Schüler ausgehend von einer realitätsnahen Handlungssituation einen typischen Softwareentwicklungsprozess durchlaufen und so ein vertieftes Verständnis für die Funktionsweise und Verwendung von Schleifen entwickeln.

KOMPETENZPROFIL

Klassenstufe:	10/11
Dauer:	5–8 Unterrichtsstunden
Lernziele:	Die Lernenden … 1. implementieren, indem sie eine bestehende Software mithilfe der for-Schleife um eine neue Funktionalität erweitern, 2. kommunizieren und kooperieren, indem sie sich gegenseitig konstruktives Feedback zu ihren Lösungsansätzen geben.
Kompetenzen:	Implementieren, Kommunizieren und Kooperieren
Themenbereiche:	Kontrollstrukturen, *Java*, zählergesteuerte Schleifen

Inhaltsfeld Algorithmen

Kontrollstrukturen in Java: Anwendung von zählergesteuerten Schleifen

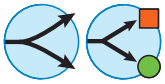
Ein Beitrag von Johann-Georg Vogelhuber

Zählergesteuerte Schleifen sind eine grundlegende Kontrollstruktur in jeder imperativen Programmiersprache. Lassen Sie Ihre Schülerinnen und Schüler ausgehend von einer realitätsnahen Handlungssituation einen typischen Softwareentwicklungsprozess durchlaufen und so ein vertieftes Verständnis für die Funktionsweise und Verwendung von Schleifen entwickeln.

KOMPETENZPROFIL

Klassenstufe:	10/11
Dauer:	5–8 Unterrichtsstunden
Lernziele:	Die Lernenden … 1. implementieren, indem sie eine bestehende Software mithilfe der for-Schleife um eine neue Funktionalität erweitern, 2. kommunizieren und kooperieren, indem sie sich gegenseitig konstruktives Feedback zu ihren Lösungsansätzen geben.
Kompetenzen:	Implementieren, Kommunizieren und Kooperieren
Themenbereiche:	Kontrollstrukturen, *Java*, zählergesteuerte Schleifen

Symbolerklärungen

	Diese Symbole markieren eine Einzel-, Partner- bzw. Gruppenarbeit.
	Diese Symbole markieren alternative Durchführungsmöglichkeiten bzw. alternative Durchführungsmöglichkeiten nach Niveaustufen.
	Tauchen diese Symbole auf, handelt es sich um binnendifferenzierte Materialien.
	Dieses Symbol markiert Materialien auf einfacherem G-Niveau bzw. Materialien eher für niedrigere Klassenstufen.
	Dieses Symbol markiert Materialien auf Normalniveau (M-Niveau).
	Dieses Symbol markiert Materialien auf höherem E-Niveau bzw. Materialien eher für höhere Klassenstufen oder Exkursmaterialien.
	Dieses Symbol markiert Hilfestellungen bzw. Tipps.
	Dieses Symbol markiert Zusatzaufgaben für schnelle Lernende.
	Dieses Symbol markiert Merkkästen und wichtige Inhalte.
	Dieses Symbol markiert am Laptop/PC zu bearbeitende Aufgaben.
	Dieses Symbol taucht auf, wenn ein Dateidownload notwendig ist.
	Dieses Symbol markiert interaktive Aufgaben zur Bearbeitung mit einem digitalen Endgerät.
	Dieses Symbol markiert das Einbinden eines Videos/Films.
	Dieses Symbol markiert eine Internetrecherche.
	Dieses Symbol taucht auf, wenn näher recherchiert werden soll oder tiefergreifende Informationen hinterlegt sind.
	Diese Symbole markieren Pro- und Kontraargumente bzw. eine Diskussion.
	Dieses Symbol markiert Aufgaben zum Ankreuzen.
	Dieses Symbol markiert Aufgaben, bei denen gerechnet werden muss.

Was sollten Sie zum Thema wissen?

Die Struktur der hier vorgestellten Unterrichtseinheit orientiert sich an einem realitätsnahen **Softwareentwicklungsprozess** mit den folgenden **Phasen:**
Analyse – **Planung** – **Implementierung** – **Test** – **Review**.
Diese einzelnen Schritte lassen sich gut in das **Modell der vollständigen Handlung** mit den folgenden **Phasen** integrieren:
Informieren – **Planen** – **Entscheiden** – **Ausführen** – **Kontrollstufe** – **Beurteilung**
Das Unterrichtskonzept soll damit dem handlungsorientierten Lernen dienen, indem es nahe an die Praxis im Berufsleben angelehnt ist. Die erworbenen Handlungskompetenzen sollen so von den Lernenden im späteren Berufsleben einfach auf die Arbeitsprozesse übertragen werden können. Daher ist dieses Modell bewusst produktorientierter als beispielsweise das Modell des Entdeckenden Lernens.
Damit die Schülerinnen und Schüler an die typische Arbeitsweise in der Softwareentwicklung herangeführt werden, ist es wichtig auf eine gewissenhafte Bearbeitung der einzelnen Schritte zu achten. Durch eine sorgfältig durchgeführte Analyse und Planung wird der kritische Schritt der Implementierung entlastet und deutlich mehr Schülerinnen und Schüler kommen zu einer funktionsfähigen Lösung.
Zentraler Bestandteil eines jeden Programmierunterrichts ist es daher, dass die einzelnen Ergebnisse der Schülerinnen und Schüler gewürdigt und die gefundenen Lösungen ausführlich besprochen werden. Durch die Kommunikation über den jeweiligen Quelltext werden Stärken und Schwächen der einzelnen Lösungen identifiziert und die Schülerinnen und Schüler können ein deutlich vertiefteres Verständnis für die Programmabläufe entwickeln.
Für eine spätere Vertiefung von agilen Entwicklungsprinzipien werden die Anforderungen an das zu erweiternde Programm in Form einer *User Story* gegeben. *User Stories* werden im Rahmen der agilen Softwareentwicklung zusammen mit Akzeptanzkriterien zur Spezifikation von Anforderungen eingesetzt. Eine *User Story* ist eine in Alltagssprache formulierte Anforderung an die Software. Für die Umsetzung einer *User Story* werden in der Regel die folgenden Schritte durchgeführt: Analyse, Planung, Umsetzung, Test und Auslieferung.

Welches Vorwissen sollten die Lernenden mitbringen?

Die Schülerinnen und Schüler sollten ein erstes Verständnis für den Aufbau von objektorientierten Programmen haben. Insbesondere müssen Klassen und Objekte erstellt und verwendet werden können. Dazu gehört die Erstellung und Verwendung von Instanzvariablen und -methoden. Weiter müssen die Schülerinnen und Schüler die grundlegenden Datentypen kennen und elementare Rechenoperationen anwenden können.
Die Unterrichtseinheit ist so aufgebaut, dass Verzweigungen und Arrays keine notwendige Voraussetzung für die Bearbeitung sind. Entsprechend der eigenen didaktischen Jahresplanung kann damit die Reihenfolge dieser Themen frei gestaltet werden. Auch die Thematisierung von weiteren Kontrollstrukturen, wie der while-Schleife kann vor oder nach dieser Unterrichtseinheit durchgeführt werden. Wird die while-Schleife vorab thematisiert, dann können bei der Umsetzung der *User Story* (**M 3**) auch Lösungen mit for- und while-Schleife miteinander verglichen und bewertet werden.

Wie kann die Erarbeitung des Themas im Unterricht erfolgen?

Vorbereitung

- Für Programmieraufgaben: eine IDE wie Visual Studio Code oder BlueJ

Für die Implementierung der *User Story* in **M 3–M 5** (siehe unten) kann eine beliebige integrierte Entwicklungsumgebung (IDE) verwendet werden. Im Unterrichtseinsatz bietet sich *BlueJ* oder *Visual Studio Code* an. Die Umgebung *BlueJ* ermöglicht ein tieferes Verständnis für den Programmablauf und einen direkten Aufruf von Methoden, während der Einsatz von *Visual Studio Code* einen deutlich größeren Realitätsbezug aufweist.

Alle notwendigen Quelltexte stehen Ihnen als Dateien zum Download für *BlueJ* oder *Visual Studio Code* zur Bearbeitung durch die Schülerinnen und Schüler zur Verfügung.

- Für Programmieraufgaben: ausreichend PC-/Laptop-Arbeitsplätze
- Für optional einsetzbare interaktive *Kahoot!*- oder *LearningSnacks*-Übungen: mobiles Endgerät

Benötigte Dateien

- Quellcode-Dateien *BlueJ*: Ordner *BlueJ* mit Unterordnern *CardioTrainer* und *SIRModell*
- Quellcode-Dateien *Visual Code Studios*: Ordner *VSCode* mit Unterordnern *CardioTrainer* und *SIR-Modell*

Einstieg

Den Einstieg in die Unterrichtseinheit bietet die Anforderungssituation „*Ein Update für die Trainings-App – Analyse des vorhandenen Quelltextes*“ in **M 1**. Die Schülerinnen und Schüler müssen zunächst gemäß der Phase Analyse die vorhandene Situation analysieren und einen Handlungsplan für die nächsten Arbeitsschritte aufstellen. Zur Unterstützung dieser Schritte enthält das Arbeitsblatt entsprechende Analysefragen. Bei der gemeinsamen Besprechung dieser Aufgaben sollte darauf geachtet werden, dass zur Ergebnisdokumentation mindestens die Erstellung eines Klassendiagramms sowie eine Vervollständigung der Kommentare im Quelltext aufgeführt werden.

Das Material **M 2** enthält die entsprechenden Schritte zur Dokumentation als einzelne Aufgaben und kann bei der Analyse des Quelltextes eingesetzt werden. Dazu sind mit den Aufgaben 3 und 4 weitere Fragen zum Verständnis des Quelltextes aufgeführt.

Um die Schülerinnen und Schüler bei der Analyse zu unterstützen, liegt der zu untersuchende Quelltext in **zwei Niveaustufen** vor. Die Variante mit bereits vorhandenen Kommentaren können leistungsschwächere Schülerinnen und Schüler über den auf **M 2** angegebenen Tipp 2 abrufen.

Erarbeitung

Die Erarbeitung der zählergesteuerten Schleife erfolgt anschließend mit der Analyse und Implementierung der *User Story* **M 3**. Die Anforderungen sind dabei ebenfalls in zwei unterschiedlichen Schwierigkeitsstufen formuliert, sodass leistungsstärkere Schülerinnen und Schüler eine etwas umfangreichere *User Story* implementieren können. Hier bietet es sich an, diesen Hinweis vor dem Austeilen des Arbeitsblattes M 3 mit auf den Weg zu geben.

Zur Planung der Umsetzung enthält das Material **M 4** detaillierte Leitfragen, die die Schülerinnen und Schüler zur Erstellung eines Handlungsplans anleiten. Diese Leitfragen können beispielsweise nach dem Handlungsmuster *Think-Pair-Share* bearbeitet werden. Eine gemeinsame Besprechung und Vervollständigung der Antworten sollte unbedingt vor der konkreten Umsetzung der *User Story* durchgeführt werden. Insbesondere sollte die Stelle der Implementierung sowie die zu verwendende Technik der for-Schleife thematisiert werden.

Die konkrete Implementierung der *User Story* erfolgt nach der Methode *Pair Programming*. Hierbei arbeiten zwei Schülerinnen bzw. Schüler gemeinsam an dem Quelltext. Diese Methode ist auf dem Arbeitsblatt **M 4** kurz beschrieben. Wichtig bei der Durchführung ist eine aktive Tätigkeit beider Teammitglieder, sodass eine produktive Kommunikation über den Quelltext entstehen kann. Zur Unterstützung bei der Implementierung kann bei Bedarf das Informationsmaterial **M 5** als Hilfe verwendet werden. Es enthält eine ausführliche Beschreibung zur for-Schleife sowie den Link zu dem folgenden Erklärvideo: *https://raabe.click/Java-Tutorial_for-Schleife*

Ergebnissicherung

Die Ergebnissicherung erfolgt nach der Implementierung in mehreren Schritten. Angelehnt an einen realitätsnahen Softwareentwicklungsprozess wird zunächst ein *Code Review* durchgeführt. Dazu begutachtet jedes Team jeweils die Lösung eines anderen Teams. **M 6** enthält hierfür eine entsprechende Situationsbeschreibung und einen Arbeitsauftrag.

Während des *Code Reviews* füllen die Schülerinnen und Schüler den Feedbackbogen **M 7** aus. Nach erfolgter Begutachtung des Quelltextes geben sich die Gruppen gegenseitig Feedback zu ihren Lösungen. Auf diese Art und Weise wird jede Lösung mindestens einmal kritisch geprüft und besprochen. An dieser Stelle können Sie nochmals auf die Regeln bei der Vergabe von Feedback hinweisen. Diese sind auch auf dem Arbeitsblatt als Hinweiskasten mit vermerkt.

Nach erfolgtem Review sollte sich eine Plenumsphase anschließen, in der einzelne Lösungen noch einmal gemeinsam besprochen werden. Hier bietet es sich an, ggf. auf fehlerhafte Lösungen einzugehen, um typische Fehlerquellen zu thematisieren.

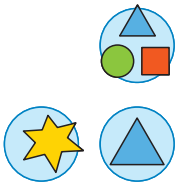
Anschließend können die fachlichen Inhalte mit den Aufgaben auf dem Arbeitsblatt **M 8** gesichert werden. Das Arbeitsblatt enthält einen Link zu einem *LearningSnack,* mit dem die Schülerinnen und Schüler interaktiv ihren Lernerfolg überprüfen können. Die Bearbeitung von M 8 kann auch gut als Hausaufgabe durchgeführt werden.

Link zum *LearningSnack*: *https://raabe.click/LearningSnack-for-Schleife*

Übung

Zur Übung des Erlernten können die differenzierten Aufgaben auf dem Arbeitsblatt **M 9** bearbeitet werden.

Optional kann auch die komplexere Projektaufgabe *„Ausbreitung von Infektionskrankheiten"* (**M 10** und **M 11**) bearbeitet werden. Diese Aufgabe erfordert ein etwas vertiefteres mathematisches Verständnis und bietet sich für leistungsstärkere Klassen oder zur Differenzierung bei besonders leistungsstarken Schülerinnen und Schülern an.



Lernerfolgskontrolle

Für die abschließende Lernerfolgsüberprüfung können Sie das Quiz auf **M 12** nutzen oder dieses interaktiv mithilfe des folgenden *Kahoot!*-Quiz durchführen lassen:

https://raabe.click/Kahoot_Java-for-Schleife

Auf einen Blick

Benötigte Materialien

☐ Für Programmieraufgaben: eine IDE wie *Visual Studio Code* oder *BlueJ*
☐ Für Programmieraufgaben: ausreichend PC-/Laptop-Arbeitsplätze
☐ Für optional einsetzbare interaktive *Kahoot!*- oder *LearningSnacks*-Übungen: mobiles Endgerät
☐ Quellcode-Dateien *BlueJ*: Ordner *BlueJ*
- Unterordner *CardioTrainer – Einstieg*: CardioTrainer.java, package.bluej
- Unterordner *CardioTrainer – MitMehrKommentaren*: CardioTrainer.java, package.bluej
- Unterordner *CardioTrainer – Lösung – Niveau_Mittel/Schwierig*: CardioTrainer.java, package.bluej
- Unterordner *SIRModell – Lösung: SIRModell.java,* package.bluej
- Unterordner *SIRModell – Hilfematerial: SIRModell.java,* package.bluej

☐ Quellcode-Dateien *Visual Code Studios*: Ordner *VSCode*
- Unterordner *CardioTrainer – Einstieg*: CardioTrainer.java
- Unterordner *CardioTrainer – MitMehrKommentaren*: CardioTrainer.java
- Unterordner *CardioTrainer – Lösung – Niveau_Mittel/Schwierig*: CardioTrainer.java
- Unterordner *SIRModell: SIRModell.java*
- Unterordner *SIRModell – Hilfematerial: SIRModell.java*

Einstieg

M 1	**Ein Update für die Trainings-App – Analyse des vorhandenen Quelltextes**
M 2	**Analyse: Bestandsaufnahme für die Klasse CardioTrainer**
Benötigt:	Cardiotrainer.java, Cardiotrainer_kommentiert.java (optional)

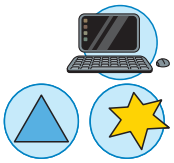
Erarbeitung

M 3	**Analyse und Implementierung: Auftrag zur Erweiterung der Klasse CardioTrainer**
M 4	**Planung und Umsetzung der *User Story***
M 5	**Infomaterial: Zählergesteuerte Schleifen in *Java***
Benötigt:	*Erklärvideo zur for-Schleife in Java https://raabe.click/Java-Tutorial_for-Schleife*

Ergebnissicherung

M 6	**Arbeitsauftrag Code Review**
M 7	**Feedbackbogen Code Review**
M 8	**Zusammenfassung for-Schleife in *Java* – Ergebnissicherung**
Benötigt:	*LearningSnack* (optional): *https://raabe.click/LearningSnack-for-Schleife*

Übung



M 9	**Aufgaben zu zählergesteuerten Schleifen**
M 10	*Projektaufgabe zur Vertiefung – Ausbreitung von Infektionskrankheiten*
M 11	**Arbeitsaufträge zur Projektaufgabe**

Lernzielkontrolle

M 12	**Teste dein Wissen!**
Benötigt:	*Kahoot!*-Quiz (optional): *https://raabe.click/Kahoot_Java-for-Schleife*

Hinweis: Alle in den Materialien genannten Codes funktionieren nur mit folgenden Anweisungen davor.

```
public class MyClass {
    public static void main(String args[]) {
Code (z.B. die for-Schleife)
    }
}
```


Inhaltsfeld Algorithmen

Kontrollstrukturen in Java: Anwendung von zählergesteuerten Schleifen

Ein Beitrag von Johann-Georg Vogelhuber

© RAABE 2021

© Sitthiphong/iStock/Getty Images Plus

Zählergesteuerte Schleifen sind eine grundlegende Kontrollstruktur in jeder imperativen Programmiersprache. Lassen Sie Ihre Schülerinnen und Schüler ausgehend von einer realitätsnahen Handlungssituation einen typischen Softwareentwicklungsprozess durchlaufen und so ein vertieftes Verständnis für die Funktionsweise und Verwendung von Schleifen entwickeln.

KOMPETENZPROFIL

Klassenstufe:	10/11
Dauer:	5–8 Unterrichtsstunden
Lernziele:	Die Lernenden … 1. implementieren, indem sie eine bestehende Software mithilfe der for-Schleife um eine neue Funktionalität erweitern, 2. kommunizieren und kooperieren, indem sie sich gegenseitig konstruktives Feedback zu ihren Lösungsansätzen geben.
Kompetenzen:	Implementieren, Kommunizieren und Kooperieren
Themenbereiche:	Kontrollstrukturen, *Java*, zählergesteuerte Schleifen