Die Hausaufgaben müssen von den Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws2223/einfuhrung-in-die-informatik/` zu berücksichtigen.

Abgabefrist ist der 26.01.2023 - 23:59 Uhr

Vorbereitung

Zur Abgabe dieser Hausaufgabe muss zunächst ein neues Respository angelegt werden. Mit dem folgenden Link kannst du das Respository für Hausaufgabe 10 anlegen bzw. einsehen.

`https://classroom.github.com/a/xMGvWsLT`

Achte bei jeder Aufgabe auf das verlangte Dateiformat. Andere Formate werden nicht akzeptiert und folglich mit **0** Punkten gewertet. Handschriftliche Abgaben werden nicht akzeptiert.

Es sollen nur die bis zur jeweiligen Übung vermittelten Konzepte verwendet werden.

Nicht oder zu spät abgegebene (Teil-)Aufgaben werden mit 0 Punkten bewertet.

Code-Formatierung

Formatiere deinen Code vor dem Abgeben wie in Vorlesung und Übung gezeigt. Verstöße gegen Formatierungs- und Namenskonventionen führen zu Punktabzug.

Lauffähigkeit

Nicht ausführbare Abgaben werden mit 0 Punkten bewertet.

3-Credit Abschnitt

Diese Hausaufgabe zählt zu dem in Übung 1 erläuterten „3-Credit Abschnitt" der Veranstaltung. Wenn du dir unsicher bist, welche Prüfung du in dieser Veranstaltung ablegen musst, informiere dich bitte bei deinem Studienservice.

Aufgabe 1 - Enum und switch case (22P)

Gegeben ist die folgende Speisekarte:

✨🍔 Burger Palace 🍔✨

Menu	contains	price
SINGLE	burger	9,99 €
SMALL	burger and fries	11,00 €
BIG	burger with fries and soft drink	11,50 €
GIGANTIC	burger with fries and nuggets, a soft drink and ice cream	12,50 €

Lege ein neues Java-Projekt an. Es soll ein Programm geschrieben werden, mit dem man eine Bestellung beim Burger Palace aufgeben kann. Es sollen drei Dateien erstellt werden, wovon zwei jeweils eine Klasse beinhalten und die dritte Datei ein Enum.

Hinweis: Bei allen Aufgaben soll das "klassische" `switch-case` verwendet werden, welches in Vorlesung/Übung gezeigt wurde!

Menu

Das Enum `Menu` soll die verschiedenen Menügrößen abbilden, die von der Speisekarte abzulesen sind.

BurgerPalace

Die Klasse `BurgerPalace` beinhaltet 5 Methoden:

- `public void printWelcome()` begrüßt den User und listet die verfügbaren Menügrößen auf. Dabei soll mithilfe von `.values()` über die Enum-Konstanten iteriert werden.
- `public void printSorry(String name)` gibt die Info aus, dass es kein Menü mit dem übergebenen Namen gibt.
- `private void printPrice(Menu menuType)` gibt aus, wie viel das Menü kostet. Die Preise sind oben in der Speisekarte gegeben. Nutze hierbei ein `switch-case`.
- `private void printItems(Menu menuType)` gibt aus, welche Bestandteile im Menü enthalten sind (siehe Speisekarte). Nutze hierfür ein `switch-case` **ohne** `break`-Anweisungen! Jedes größere Menü ist eine Obermenge des nächst kleineren Menüs, sodass der „fall through"-Mechanismus genutzt werden kann und soll.
- `public void orderBurger(Menu menuType)` nutzt die Hilfsmethoden, um die Items und den Preis des Menüs auszugeben und printet abschließend ein kurzes Dankeschön für die Bestellung.

Main

Die Klasse `Main` beinhaltet die `main`-Methode. Darin wird eine neue Instanz des `BurgerPalace` erstellt, um im weiteren Verlauf die verfügbaren Methoden darauf aufrufen zu können.

Der User soll per Kommandozeilenargument die gewünschte Menügröße eingeben können, sodass der Burger beim BurgerPalace bestellt werden kann (`orderBurger()`). Wurde kein Kommandozeilenargument übergeben, wird `printWelcome` aufgerufen.

Das Kommandozeilenargument muss mithilfe von `.valueOf()` vom Typ `String` zu `Menu` umgewandelt werden, damit anschließend `orderBurger()` aufgerufen werden kann. Dabei muss die mögliche `IllegalArgumentException` durch ein `try-catch` gefangen werden. Im Fehlerfall wird mittels `printSorry` ein Fehlertext ausgegeben.

Beispiel-Ablauf

keine Kommandozeilenargumente:

```
Welcome to BurgerPalace!
You can order a burger. We offer following menus to extend your order:
SINGLE
SMALL
BIG
GIGANTIC
Choose wisely!
```

Kommandozeilenargumente: `SMALL`

```
Your SMALL menu costs 11,00 €
Your ordered meal will include:
- Fries
- Burger
Thanks for your order!
```

Kommandozeilenargumente: `MINI`

```
We are very sorry, we don't offer a menu called "MINI"
```

Abgabe

Gib deine Dateien (`Main.java, BurgerPalace.java, Menu.java`) mit der Lösung im GitHub-Repository zur aktuellen Hausaufgabe ab.

Aufgabe 2 - Code testen (14P)

Dependencies und Projektstruktur

Erstelle für diese Aufgabe ein neues Java-Projekt. Zunächst müssen die benötigten Dependencies heruntergeladen werden, um JUnit verwenden zu können. Die benötigten Links findest du hier unter „Plain-old JAR":

`https://github.com/junit-team/junit4/wiki/Download-and-Install#plain-old-jar`

Anschließend müssen die zwei heruntergeladenen `.jar`-Dateien zum Projekt hinzugefügt werden. Danach muss noch das Verzeichnis für die Tests erstellt und markiert werden. Halte dich dafür an die Vorgehensweise aus Vorlesung / Übung.

Tests schreiben

Lade dir die Datei `PrimeChecker.java` aus deinem Hausaufgaben-Repository herunter und füge sie in dein `src`-Verzeichnis ein. Die Klasse enthält die Methode `isPrimeNumber`, welche einen Integer bekommt und `true` zurückgibt, wenn der übergene Integer eine Primzahl ist und `false` zurückgibt, wenn der übergebene Integer keine Primzahl ist. Nun soll getestet werden, ob die Methode wirklich korrekt funktioniert.

Lege nun eine Testklasse mit dem Namen `PrimeCheckerTest.java` an. In dieser sollen Testmethoden geschrieben werden, die folgende Fälle testen:

- die Methode `testIsPrimeNumberIsPrime` testet die Eingabe von Primzahlen
- die Methode `testIsPrimeNumberIsNotPrime` testet die Eingabe von natürlichen Zahlen, die keine Primzahlen sind
- die Methode `testIsPrimeNumberZero` testet die Eingabe der Zahl 0
- die Methode `testIsPrimeNumberNegative` testet die Eingabe von negativen Zahlen

Somit enthält die Testklasse schließlich diese **4** Test-Methoden. In jeder Test-Methode muss zunächst ein Objekt der Klasse `PrimeChecker` angelegt werden, um dann die gegebene Methode `isPrimeNumber` mit den entsprechenden Werten aufrufen zu können. Im Anschluss muss der Rückgabewert mittels asserts übergeprüft werden.

Hinweis: Eine Primzahl ist eine natürliche Zahl, die größer als 1 und ausschließlich durch sich selbst und durch 1 teilbar ist.

Tests auswerten

Mithilfe deiner geschriebenen Tests soll herausgefunden werden, an welchen Stellen die gegebene Methode fehlerhaft ist. Wenn du deine Tests fertig programmiert hast, kannst du bei fehlschlagenden Tests darauf schließen, an welcher Stelle die Logik der `isPrimeNumber()`-Methode fehlerhaft ist. Schau dir jetzt den gegebenen Programmcode an, um festzustellen, wo genau der Fehler liegt und was verbessert werden muss. Beschreibe die Ergebnisse deiner Auswertung in einem Kommentar über der jeweiligen fehlschlagenden Testmethode.

Abgabe

Gib deine Lösung in einer Datei mit dem Namen `PrimeCheckerTest.java` im GitHub-Repository zur aktuellen Hausaufgabe ab. Die Klasse `PrimeChecker` darf verändert werden, muss allerdings nicht abgegeben werden.