Pflichtaufgabe 06.12.2024 – Exceptions

Aufgabe 1 – ca. 90 Minuten

Bitte nehme dir die Zeit und verbessere dein Englisch.

Alternativ/Zusätzlich: Nutze die Zeit und beginne damit die Fragen im kleinen „blauen Buch“ (Pursley) durchzugehen.

Aufgabe 2 - Restaurantbestellung mit bestehenden Exceptions

- **Teil 1: Vererbung**
- **Gericht (abstrakte Klasse)**:
    - Erstelle eine abstrakte Klasse *Gericht*, die folgende Eigenschaften und Methoden enthält:
        - String name (Name des Gerichts, z. B. "Pizza Margherita", "Spaghetti Bolognese").
        - double preis (Preis des Gerichts).
        - Ein Konstruktor, der name und preis setzt.
        - Eine abstrakte Methode berechnePreis(), die den Preis des Gerichts zurückgibt. (Für spätere Erweiterungen könnte hier ein Rabattsystem eingebaut werden, aber das ist optional).
- **Pizza (Unterklasse von Gericht)**:
    - Erbt von der abstrakten Klasse *Gericht*.
    - Füge eine Methode getPizzaInfo() hinzu, die den Namen, den Preis und eine Beschreibung des Gerichts zurückgibt.
- **Pasta (Unterklasse von Gericht)**:
    - Erbt von der abstrakten Klasse *Gericht*.
    - Füge eine Methode getPastaInfo() hinzu, die den Namen, den Preis und eine Beschreibung des Gerichts zurückgibt.
- **Getränk (Unterklasse von Gericht)**:
    - Erbt von der abstrakten Klasse *Gericht*.
    - Füge eine Methode getGetraenkInfo() hinzu, die den Namen, den Preis und eine Beschreibung des Getränks zurückgibt.

- **Teil 2: Exception-Handling**
- **IllegalArgumentException**:
    - Wenn beim Hinzufügen eines Gerichts zu einer Bestellung der Preis <= 0 ist, soll eine IllegalArgumentException geworfen werden. Die Nachricht könnte lauten: "Der Preis eines Gerichts muss positiv sein.".
    - Wenn ein Gericht zum zweiten Mal bestellt wird, wirf eine IllegalArgumentException mit der Nachricht: "Gericht wurde bereits bestellt.".
- **NullPointerException**:
    - Wenn beim Erstellen eines Gerichts der Name oder Preis null oder ungültig (z. B. <= 0) ist, wirf eine NullPointerException mit der Nachricht: "Gericht muss einen Namen und einen positiven Preis haben.".
- **NoSuchElementException**:
    - Wenn ein Gericht aus der Bestellung entfernt werden soll, aber nicht vorhanden ist, wirf eine NoSuchElementException mit der Nachricht: "Gericht nicht in der Bestellung gefunden.".
- **InputMismatchException**:
    - Wenn der Benutzer eine ungültige Zahl (z. B. ein Textwert anstelle einer Zahl) für den Preis oder die Anzahl der Bestellungen eingibt, soll eine InputMismatchException geworfen werden. Die Nachricht könnte lauten: "Ungültige Eingabe. Bitte geben Sie eine Zahl ein.".
- **ArithmeticException**:
    - Wenn der Betrag für die Zahlung negativ ist oder der Benutzer einen zu niedrigen Betrag eingibt, soll eine ArithmeticException geworfen werden. Die Nachricht könnte lauten: "Der Betrag muss positiv sein.".
- **Teil 3: Restaurantverwaltung und Anwendung**
- **Restaurant**:
    - Erstelle eine Klasse Restaurant, die eine Liste von bestellten Gerichten verwaltet.
    - Die Klasse sollte folgende Methoden haben:
        - gerichtBestellen(Gericht gericht): Fügt ein Gericht zur Bestellung hinzu. Wenn das Gericht bereits bestellt wurde oder der Preis ungültig ist, wird eine IllegalArgumentException geworfen.

        - gerichtEntfernen(Gericht gericht): Entfernt ein Gericht aus der Bestellung. Wenn das Gericht nicht vorhanden ist, wird eine NoSuchElementException geworfen.
        - getBestellungInfo(): Gibt die gesamte Bestellung zurück, einschließlich der Namen und Preise der Gerichte.
        - berechneRechnung(): Berechnet die Gesamtsumme der Bestellung.

- **Main-Methode**:
    - Erstelle ein Restaurant und füge verschiedene Gerichte (Pizza, Pasta, Getränk) hinzu.
    - Versuche, Gerichte zu bestellen, zu entfernen, die Rechnung zu berechnen und Fehler wie IllegalArgumentException, NoSuchElementException, ArithmeticException, etc. zu behandeln.