# Übungsblatt: Prüfungsvorbereitung

## Übungsaufgabe 1: Klassen, Collections & File I/O

*Themen: Szenario Restaurant, ArrayList/Vector, Exception Handling*

### A: Die Bestellung

Schreiben Sie eine Klasse `Bestellung`. Jede Bestellung verfügt über folgende Attribute:

- Einen `Namen` (des Gerichts, Typ String)
- Einen `Preis` (Typ double)
- Eine `TischNummer` (Typ int)

### B: Die Verwaltung

Schreiben Sie eine Klasse `Kueche`.

- Diese nutzt intern einen **Vektor** (oder eine `ArrayList`), um Objekte vom Typ `Bestellung` zu speichern.
- Implementieren Sie eine Methode:
  `public void neueBestellung(String name, double preis, int tisch)`
  Diese Methode soll ein neues Objekt erzeugen und in die Liste einfügen.

### C: Dateiverarbeitung (File I/O)

Implementieren Sie in der Klasse `Kueche` eine Methode:
`public void tagesAbschlussSpeichern(String dateiname)`

- Die Methode soll alle aktuellen Bestellungen in eine Textdatei schreiben.
- Formatbeispiel in der Datei: `"Tisch 1: Pizza - 9.50 Euro"`
- Fangen Sie mögliche Fehler (Exceptions) mit einem Standard `try-catch`-Block ab (keine eigene Exception-Klasse schreiben!).

---

## Übungsaufgabe 2: Parallele Programmierung

*Themen: Threads, Runnable, sleep*

Schreiben Sie eine Klasse `Koch`, die von `Thread` erbt (oder `Runnable` implementiert).

1. Ein Koch wird mit einem `GerichtNamen` (String) und einer `ZubereitungsZeit` (int, in Millisekunden) initialisiert.
2. In der `run()`-Methode soll der Koch folgende Schritte ausführen:
   - Konsolenausgabe: `"Fange an mit [Gericht]..."`
   - Schlafen für die angegebene Zeit (`Thread.sleep()`).

- Konsolenausgabe: `"[Gericht] ist fertig!"`

3. **Main-Methode:** Erstellen Sie in einer Hauptklasse drei Koch-Objekte mit unterschiedlichen Zeiten und starten Sie diese so, dass sie **parallel** arbeiten.

# Übungsaufgabe 3: Rekursion

*Themen: Rekursive Aufrufe, Logik, Konsolenausgabe*

### A: Die Quersumme

Schreiben Sie eine **rekursive** Funktion `int quersumme(int n)`.
Beispiel: `quersumme(123)` $\rightarrow 1 + 2 + 3 = 6$.
*Tipp: Nutzen Sie Modulo 10 und Ganzzahldivision durch 10.*

### B: Rekursives Muster

Schreiben Sie eine rekursive Methode `printPattern(int n)`, die folgende Ausgabe auf der Konsole erzeugt (Beispiel für $n = 4$):

```
*
**
***
****
***
**
*
```

*Hinweis: Die Methode ruft sich selbst in der Mitte auf. Reihenfolge: Print, rekursiver Aufruf, Print.*

---

# Übungsaufgabe 4: Grafik & Logik

*Themen: JPanel, paintComponent, Mathe-Logik*

Erstellen Sie eine Klasse `SternenPanel`, die von `JPanel` erbt.

### A: Die Datenhaltung

- Das Panel besitzt eine Liste (Vektor oder ArrayList) von `Point`-Objekten.
- Erzeugen Sie im Konstruktor 50 zufällige Punkte $(x, y)$ innerhalb der Panel-GröSSe und speichern Sie diese in der Liste.

### B: Das Zeichnen (paintComponent)

Überschreiben Sie die `paintComponent(Graphics g)` Methode:

1. Zeichnen Sie den Hintergrund schwarz.
2. Zeichnen Sie jeden Punkt aus der Liste als kleinen weiSSen Kreis.
3. **Die Logik:** Überprüfen Sie in einer doppelten Schleife den Abstand aller Punkte zueinander.
4. Wenn der Abstand zwischen Punkt A und Punkt B kleiner als 50 Pixel ist, zeichnen Sie eine **gelbe Linie** zwischen ihnen.

*Formel für die Distanz:*

$$\text{Distanz} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

(Alternativ nutzen Sie `Point.distance(...)`)

---

## Übungsaufgabe 5: Theorie-Check

Beantworten Sie folgende Fragen stichpunktartig:

1. **Vektor vs. Array:** Nennen Sie zwei wesentliche Unterschiede zwischen einem primitiven Array (`[]`) und der Klasse `Vector`.
2. **Packages:** Wozu dienen Packages in Java? Nennen Sie einen Vorteil.
3. **Exceptions:** Was ist der Unterschied zwischen einer *Checked Exception* (z.B. `IOException`) und einer *Unchecked Exception* (z.B. `NullPointerException`) in Bezug auf den Compiler?
4. **Vererbung:** Kann eine Klasse von mehreren abstrakten Klassen erben? Kann sie mehrere Interfaces implementieren?