Projekt: Simulation eines Smart-Parking-Systems (CPS)

In diesem Projekt entwickeln Sie eine Simulation eines intelligenten Parkhauses (Smart Parking System) mithilfe der Programmiersprache Python. Ziel ist es, den realistischen Betrieb eines Parkhauses softwareseitig nachzubilden – vom Einfahren eines Fahrzeugs über die Bezahlung bis hin zur Ausfahrt.

Das Parkhaus stellt ein cyber-physisches System (CPS) dar:

- Die physische Ebene besteht aus begrenzten Parkplätzen und Geld.
- Die cyber-Ebene überwacht diese Zustände, trifft Entscheidungen und steuert den Ablauf.

Das System wird vollständig textbasiert umgesetzt, orientiert sich aber an realen Abläufen eines echten Parkhauses.

Durch dieses Projekt lernen Sie:

- strukturierte Programmierung mit Python
- Umgang mit Benutzereingaben und Fehlereingaben
- Modellierung von Zuständen und Ressourcen
- Arbeiten mit Funktionen und Modulen
- Trennung von Benutzer- und Wartungsfunktionen
- Grundlagen cyber-physischer Systeme
- Persistenz von Daten (Speichern/Laden)

Szenario: Das Parkhaus

Das Parkhaus besitzt:

- eine begrenzte Anzahl an Parkplätzen
- einen aktuellen Belegungszustand
- Einnahmen durch Parkgebühren

Fahrzeuge können:

- einfahren
- parken
- ausfahren
- bezahlen

Wartungspersonal kann:

- das Parkhaus leeren
- Einnahmen entnehmen
- einen Statusbericht abrufen

Anforderungen an das System

1. Benutzerinteraktion (Hauptmenü): **(5 Punkte)**
Das Programm zeigt dem Benutzer ein Menü mit folgenden Optionen:

- `enter` – Fahrzeug fährt ins Parkhaus
- `exit` – Fahrzeug verlässt das Parkhaus
- `status` – aktueller Status des Parkhauses
- `maintenance` – Wartungsmodus
- `off` – System ausschalten

Nach jeder Aktion kehrt das System ins Hauptmenü zurück.

2. Parkplätze (physische Ressource): **(4 Punkte)**

- Das Parkhaus hat z. B. 10 Parkplätze
- Beim Einfahren:
  - Wird geprüft, ob noch freie Plätze vorhanden sind
  - Falls nicht → Meldung: „Parkhaus voll“
- Beim Ausfahren:
  - Ein Parkplatz wird wieder freigegeben

3. Parkdauer & Gebühren: **(4 Punkte)**

- Beim Einfahren wird die Einfahrtszeit gespeichert
- Beim Ausfahren wird:
  - die Parkdauer berechnet
  - der zu zahlende Betrag ermittelt (z. B. 2 € pro Stunde)
- Angebrochene Stunden zählen als volle Stunden

4. Bezahlvorgang: **(4 Punkte)**

- Der Benutzer muss die Parkgebühr bezahlen
- Erlaubte Münzen: 0,50 € | 1 € | 2 €
- Das System prüft:
  - ob genug bezahlt wurde
  - ob Wechselgeld notwendig ist
- Annahme: Das Parkhaus hat unendlich Wechselgeld

5. Statusbericht (`status`): **(3 Punkte)**
Der Statusbericht zeigt:

```
--- PARKHAUS STATUS ---
Freie Parkplätze: 4
Belegte Parkplätze: 6
Einnahmen: 24,00 €
----------------------
```

6. Wartungsmodus: **(8 Punkte)**
Der Wartungsmodus ist nicht für normale Benutzer gedacht.

Zugriff

- Aktivierung über den Befehl “maintenance”
- Optional: Codewort erforderlich

Befehle im Wartungsmodus

- `empty` – alle Fahrzeuge entfernen (Parkhaus leeren)
- `take money` – Einnahmen entnehmen
- `report` – detaillierter Statusbericht
- `exit` – Wartungsmodus verlassen

Nach jeder Wartungsaktion muss der Systemzustand gespeichert werden.

7. Persistenz (Speichern & Laden): **(4 Punkte)**

- Der Zustand des Parkhauses wird in einer CSV-Datei gespeichert:
    - Anzahl belegter Parkplätze
    - Einnahmen

- Beim Start des Programms:
    - Wird versucht, den letzten Zustand zu laden
- Beim Beenden (`off`):
    - Wird der aktuelle Zustand gespeichert

8. Strukturvorgaben: **(6 Punkte)**
Das Projekt soll auf mehrere Dateien aufgeteilt werden, z. B.:

- `config.py` – feste Werte (Parkplätze, Gebühren, Codewort)
- `storage.py` – Laden/Speichern (CSV)
- `payment.py` – Bezahlvorgang
- `parking.py` – Parklogik (ein-/ausfahren)
- `utils.py` – Hilfsfunktionen
- `main.py` – Programmablauf

9. Erweiterungen (optional) **(Zusatz: 10 Punkte)**

- Zeitverzögerungen bei Ein- und Ausfahrt (`time.sleep`)
- Logging aller Ereignisse (Ein-/Ausfahrten)
- Unterschiedliche Tarife (Tag/Nacht)
- Statistik: durchschnittliche Parkdauer
- Simulation mehrerer Parkhäuser

Dieses Projekt ist ein vollwertiges Beispiel für ein cyber-physisches System, da Software reale Zustände (Parkplätze, Zeit, Geld) überwacht, Entscheidungen trifft und aktiv eingreift. Es ist bewusst realitätsnah gestaltet und bildet typische Abläufe industrieller Systeme ab.

Abgabe der Lösung (spätestens 11.02.26, 19:00 Uhr)

Die Abgabe dieses Projekts erfolgt über ein öffentliches GitHub-Repository.

Anforderungen an die Abgabe **(4 Punkte)**

- Erstellen Sie ein öffentliches Repository auf GitHub.
- Der Repository-Name soll sinnvoll gewählt sein, z. B.:
  - smart-parking-cps
  - smart-parking-system
- Laden Sie den vollständigen Quellcode hoch
- Das Projekt muss lauffähig sein (`python main.py`).

Ihr Repository muss eine Datei **`README.md`** enthalten, die mindestens folgende Punkte beschreibt: **(4 Punkte)**

- kurze Projektbeschreibung
- Erklärung, warum es sich um ein cyber-physisches System handelt
- Übersicht der Projektstruktur (Dateien und Aufgaben)
  Anleitung zum Starten des Programms
- optionale Erweiterungen (falls umgesetzt)