Projektarbeit Programmieren

> Python - Spiel: Verlassene Raumstation

Allgemein

Abgabe

- Einzelarbeit, keine Gruppenabgabe!
- Die Abgabe erfolgt per Moodle
- Jeder muss die Projektarbeit einzeln hochladen
- Deadline der Abgabe: `31.03.2025 - 23:59`

Aufgabenstellung

- Aufgabe der Projektarbeit ist es, ein konsolenbasiertes Spiel in Python zu implementieren.
- Zusätzlich muss eine Dokumentation abgegeben werden, in der Quellcode, Spiellogik und Metriken vorgestellt werden.

Verlassene Raumstation – Spielregeln:

Spielfeld

Das Spielbrett stellt verschiedene Bereiche einer verlassenen Raumstation dar, auf denen sich versteckte Gefahren befinden, z.B. mechanische Fallen.

Aufdecken

Jeder Zug besteht darin, einen Bereich zu "scannen". Ist der Bereich ungefährlich, erscheint eine Zahl, die anzeigt, wie viele benachbarte Bereiche Fallen enthalten.

Spielziel

Das Ziel ist, alle sicheren Bereiche aufzudecken, ohne dabei einen Bereich mit einer Gefahr zu aktivieren. Wird ein gefährlicher Bereich aufgedeckt, endet das Spiel sofort.

Anforderungen

Technologie

- Es muss ein Konsolenspiel implementiert werden.
- Das Programm muss in der Programmiersprache Python entwickelt werden.
- Das Programm muss in Python 3.13.1 unter Linux (Ubuntu 20.04 LTS) lauffähig sein.
- Das Programm muss eine nutzerfreundliche Kommandozeilen-Schnittstelle bieten.
- Folgende Spiel-Module sind zulässig: `os`, `io`, `sys`, `random`, `unittest`.
- Folgende Bibliotheken sind zulässig: `pylint`, `coverage`, `mypy`.

Funktionale Anforderungen

- Zu Spiel-Beginn soll ein zufälliges neues Spielbrett angelegt werden.
- Das Spielbrett enthält zufällig verteilte freie Felder und Fallen-Felder.
- Die Mindestgröße des Spielbretts soll 5x5 Felder groß sein.
- Das *verdeckte* Spielfeld muss auf der Kommandozeile angezeigt werden.
- Der Spieler muss ein Feld "scannen" dürfen. Wird:
  - eine Falle "gescannt" endet das Spiel sofort.
  - ein freies Feld "gescannt" wird eine Zahl eingeblendet, die die Summe der seitlich und diagonal benachbarten Fallen anzeigt.

Nicht-funktionale Anforderungen

- Zuverlässigkeit:
  - Stabile Funktionalität ohne Abstürze.
  - Fehlertolerante Handhabung von ungültigen Eingaben.
- Benutzerfreundlichkeit:
  - Einfache Bedienung auch für Benutzer ohne technische Vorkenntnisse.

Robuste Software-Entwicklung

- Es müssen automatisierte Unittests entwickelt werden.
- Es muss die Bibliothek `unittest` genutzt werden.
- Pro Datei muss eine Testabdeckung von 75% erreicht werden (Prüfung über `coverage`).
- Die Tests müssen logisch sinnvoll entwickelt sein.

Statische Code-Analyse - Pylint

- Es muss `pylint` auf jede Python Datei ausgeführt werden.
- Alle Pylint Warnungen müssen behoben werden.
- Meldungen können durch eine gute Begründung unterdrückt werden.
- Keine Dokumentation der Test-Dateien nötig (`#pylint: disable=C`).
- Maximal 5 Personalisierungen sind erlaubt, diese sind in `pylintrc` zu dokumentieren, zu testen und mit abzugeben

Type Checker - MyPy

- Alle Python Dateien müssen von mypy überprüft werden.
- Es müssen alle mypy Warnungen behoben werden.
- Es muss folgende `mypy.ini` Datei genutzt werden

```
[mypy]
warn_return_any = True
warn_unused_configs = True
disallow_untyped_defs = True
disallow_untyped_calls = True
disallow_incomplete_defs = True
```

Technische Anforderungen

- Alle Bibliotheken müssen in der Datei `requirements.txt` verwaltet werden.
- Die Ordnerstruktur des Projektes muss folgendermaßen aussehen:

```
.
├── mypy.ini
├── README.md
├── requirements.txt
├── documentation
│   ├── documentation.pdf
├── source
│   ├── main.py
│   └── example.py
└── tests
    ├── test_main.py
    └── test_example.py
```

Dokumentation

- Die Dokumentation muss als PDF abgegeben werden.
- Es müssen alle Funktionen des Spiels und der grundlegende Aufbau beschrieben werden.
- Es muss die Version aller genutzten Bibliotheken dokumentiert sein.
- Es muss eine Architekturbeschreibung vorhanden sein.
- Es muss das Nutzerinterface beschrieben sein.
- Es muss ein Programmablauf dokumentiert sein.
- Es müssen die Ergebnisse aller statischen und dynamischen Tools vorhanden sein (Unittests, Coverage, Pylint, MyPy).

Bewertung

- Die Bewertung erfolgt über die Excel Tabelle.
- Jeder muss eine Selbsteinschätzung vornehmen.