Лабораторная работа №4. Симуляция с пользовательскими коллекциями и псевдослучайной моделью

Обязательные требования

1. **Реализовать не менее двух пользовательских коллекций:**
   - **одну списковую**, через композицию,
   - **одну словарную**
2. Коллекции должны поддерживать:
   - `__iter__` — итерирование,
   - `__len__` — длину,
   - `__getitem__` — доступ по индексу / ключу,
   - поддержку срезов (для списковой коллекции),
   - добавление и удаление элементов.
3. В словарной коллекции обязательно должна быть **логика, связанная с предметной моделью** (индексация книг / хранение балансов игроков / доходов гусей и т.п.).
4. В проекте должны быть как минимум:
   - **один базовый класс**,
   - **два производных класса**, расширяющих или переопределяющих поведение.
5. Должен быть реализован **магический метод**, отражающий предметную область:
   - `__add__`, `__call__`, `__repr__`, `__contains__` — на выбор, но со смыслом.
6. Должна быть реализована **функция симуляции**:

```
def run_simulation(steps: int = 20, seed: int | None = None) -> None:
    ...
```

Функция:

- выполняет симуляцию в `steps` шагов,
- использует `random`,
- при заданном `seed` генерирует **идентичную** последовательность событий.

7. На каждом шаге симуляции происходит **одно случайное событие**, выбранное из списка.
8. Все события симуляции должны **выводиться в консоль**, в формате текстового лога.

Вариант 1. «Библиотека»

Требуется реализовать систему управления библиотекой.

Реализовать:

1. Класс `Book`

- обязательные поля: `title`, `author`, `year`, `genre`, `isbn`.

2. Пользовательская списковая коллекция `BookCollection`

- содержит список книг,
- должна поддерживать:
  - `__getitem__` (индексы и срезы),
  - `__iter__`,
  - `__len__`,
  - методы добавления/удаления.

3. Пользовательская словарная коллекция для индексов

- выполняет индексацию книг по:
  - ISBN,
  - автору,
  - году.
- изменение коллекции должно отражаться в индексах.

4. Класс `Library`

В библиотеке должны быть:

- коллекция всех книг (`BookCollection`),
- коллекция индексов (`IndexDict`),
- методы для поиска книг.

Методы поиска должны использовать пользовательские коллекции, а не raw-list.

5. Псевдослучайная симуляция

На каждом шаге случайно выполняется одно событие:

- добавление новой книги,
- удаление случайной книги,
- поиск по автору/жанру/году,
- обновление индекса,
- попытка получить книгу, которой нет (проверка обработки).

События должны быть **разнообразными** — минимум 5 разных типов.

Вариант 2. «Казино и Гуси».

Требуется реализовать игровую симуляцию казино, в которой участвуют игроки, гуси и денежные потоки.

Реализовать:

1. Классы предметной области:

- `Player` — игрок, с полем `balance`,
- `Goose` — базовый гусь (имя, honk_volume),
- два подкласса:
  - `WarGoose` — атакует игроков,
  - `HonkGoose` — Обладает уникальным навыком «крик», когда его вызывают,
- `Chip` — фишки казино, с `__add__`.

2. Пользовательская словарная коллекция:

- `CasinoBalance` или аналог — имя игрока → баланс,
- изменения балансов должны логироваться (например, переопределение `__setitem__`).

3. Пользовательская списковая коллекция:

- `PlayerCollection`, `GooseCollection` или `ChipCollection`,
- поддерживает индексацию, срезы, итерацию.

4. Класс `Casino`

Содержит:

- коллекцию игроков,
- коллекцию гусей,
- коллекции балансов и доходов гусей.

Методы (минимум):

- регистрация игрока/гуся,
- выполнение одного шага симуляции,
- выполнение атаки/крика/ставки/кражи.

5. Псевдослучайная симуляция

На каждом шаге выбирается случайное действие, например:

- игрок делает ставку,
- игрок выигрывает/проигрывает,
- WarGoose атакует игрока,
- HonkGoose кричит (влияет на баланс),
- гусь пытается украсть деньги,
- объединение гусей в стаю через `__add__`,
- случайный игрок проигрывает всё из-за паники и т.п.

Минимум **5 различных** случайных событий.

Критерии оценивания

1. Корректность пользовательских коллекций (0–30)

- списковая коллекция корректно поддерживает индексы, срезы, итерацию (10)
- словарная коллекция корректно отражает изменения (10)
- магические методы коллекций работают корректно (5)
- коллекции сохраняют корректное поведение в ходе симуляции (5)

2. Корректность предметной модели (0–20)

- основные действия сущностей работают корректно (10)
- наследование используется для различий поведения (5)
- магические методы предметной области работают корректно (5)

3. Корректность псевдослучайной симуляции (0–25)

- симуляция выполняет случайные события (10)
- используется random + seed (5)
- события корректно изменяют состояние объектов/коллекций (5)
- лог симуляции отображается в консоль (5)

4. Взаимодействие коллекций и симуляции (0–15)

- состояние объектов корректно отражается в коллекциях (5)
- коллекции используются в логике случайных событий (5)
- нет невозможных состояний (5)

5. Индексация и поиск (0–10)

- индексирование через пользовательские коллекции работает корректно (5)
- поиск по параметрам/ключам даёт корректные результаты (5)