Лабораторна робота 2

Метадані

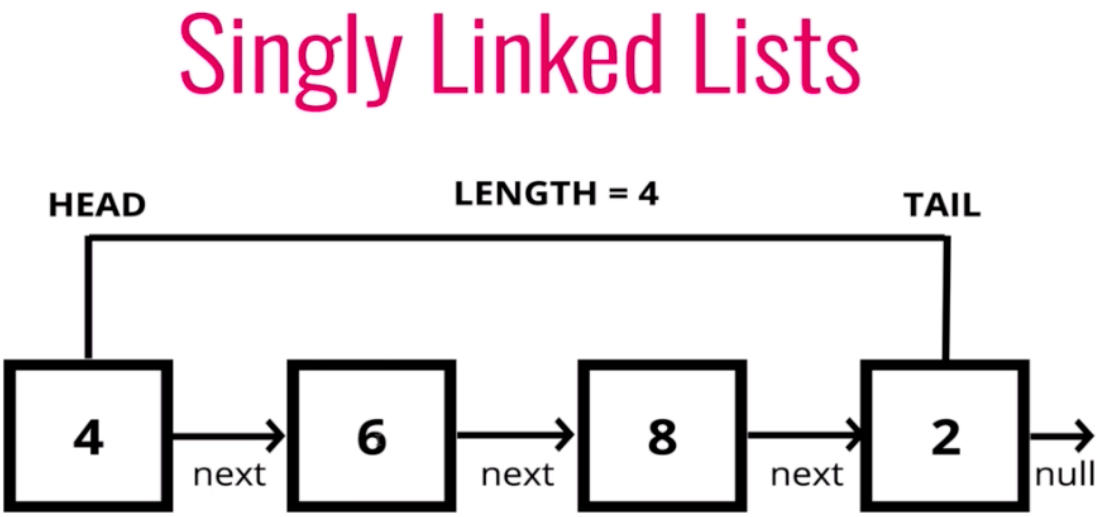
Тема	Unit tests. CI
Мета	Навчитися покривати код проекту unit-тестами.
Дедлайн	2022-03-01 10:25 Europe/Kyiv

Завдання

Застосунок

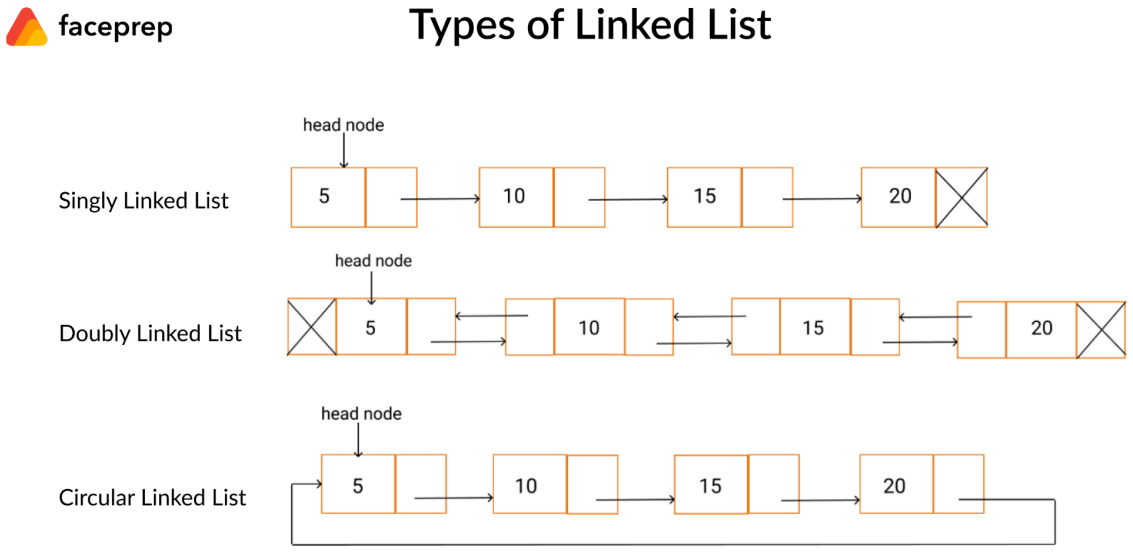
Необхідно реалізувати застосунок, який демонструє роботу із типізованим зв’язним списком згідно варіанту.

Зв’язаний список — це структура даних, яка складається із вузлів, зв’язаних між собою. Вузол містить в собі безпосередньо дані та інформацію про сусідні вузли.



Залежно від кількості зв’язків у списку розрізняють однобічно зв'язані та двобічно зв’язані списки.

У однобічно зв’язаному списку кожен вузол містить інформацію про себе (дані) та про наступний елемент (посилання / зв’язок). У двобічно зв’язаному списку кожен вузол містить дані та посилання на наступний ТА попередній елементи.



Якщо останній елемент списку у якості наступного елементу містить посилання на перший, то такий список називають кільцевим.
Докладнішу інформацію про списки можна знайти на Wikipedia.

Необхідно написати клас, який реалізує список згідно варіанту, елементами якого будуть літери (Character). Клас списку повинен підтримувати наступні методи:

- Операцію визначення довжини списку. Якщо список непорожній, то ця операція повинна повертати кількість елементів у списку. Якщо список порожній, то ця операція повинна повертати 0.
  `def length() -> Integer`
- Операцію додавання елементу в кінець списку.
  `def append(element: Character) -> None`
- Операцію вставки елементу на довільну позицію у списку. Нумерація елементів списку починається з 0.
  `def insert(element: Character, index: Integer) -> None`
  У випадку передачі некоректного значення позиції (наприклад, від'ємне число, або число, більше за кількість елементів у списку) метод повинен генерувати виключну ситуацію
- Операцію видалення елементу зі списку на вказаній позиції. Метод повинен повертати значення того елементу, який видаляється. Нумерація елементів списку починається з 0.
  `def delete(index: Integer) -> Character`
  У випадку передачі некоректного значення позиції (наприклад, від'ємне число, або число, більше за індекс останнього

елементу списку) метод повинен генерувати виключну ситуацію

-

- Операцію видалення елементів зі списку за значенням. Метод видаляє зі списку усі елементи, які за значенням відповідають шуканому.

  `def deleteAll(element: Character) -> None`

  У випадку передачі елемента, який у списку відсутній, жодні зміни до списку не застосовуються.
- Операцію отримання елементу списку на довільній позиції

  `def get(index: Integer) -> Character`

  У випадку передачі некоректного значення позиції (наприклад, від'ємне число, або число, більше за індекс останнього елементу списку) метод повинен генерувати виключну ситуацію
- Операцію копіювання списку. При виклику повинен створити копію поточного списку та повернути її.

  `def clone() -> List`
- Операцію обернення списку. Метод повинен змінити порядок елементів у поточному списку задом наперед. Елемент, що був останнім стане першим, передостаннім — другим, … а перший — останнім.

  `def reverse() -> None`
- Операцію пошуку елемента за значенням з голови списку. Метод повинен знайти перший елемент у списку, що дорівнює шуканому та повернути його позицію. Нумерація елементів списку починається з 0. У випадку відсутності шуканого елемента у списку, метод повертає -1

```
def findFirst(element: Character) -> Integer
```

- Операцію пошуку елемента за значенням з хвоста списку. Метод повинен знайти останній елемент у списку, що дорівнює шуканому та повернути його позицію. Нумерація елементів списку починається з 0. У випадку відсутності шуканого елемента у списку, метод повертає -1.

  ```
  def findLast(element: Character) -> Integer
  ```
- Операцію очищення списку. Метод видаляє усі елементи списку.

  ```
  def clear() -> None
  ```
- Операцію розширення списку. Метод приймає інший список та додає до поточного списку усі елементи останнього. При цьому подальші зміни в другий список не повинні впливати на перший.

  ```
  def extend(elements: List) -> None
  ```

Програма повинна містити демонстрацію використання усіх методів класу (у довільному порядку).

Варіант

Варіант визначається остачею від ділення номера залікової книжки на 2:

остача = 0: однобічно зв’язаний кільцевий список

остача = 1: двобічно зв'язаний список

Обов’язкові вимоги

- Вихідний код застосунку повинен бути розміщений на GitHub

- Заборонено користуватися вбудованими колекціями та бібліотеками колекцій (списки необхідно реалізовувати самостійно)
- Поведінка класу списку, який вами реалізований у рамках даної лабораторної роботи повинна бути покрита unit-тестами.
- В репозиторії на GitHub повинна бути налаштована CI GitHub Actions (також допускається використання інших CI систем), яка запускає тести на кожен коміт в основній гілці репозиторія.
- Репозиторій повинен містити принаймні один коміт, на якому впали тести на CI. (демонстрація того, що ваші тести можуть падати, і що CI це відловлює).
- Репозиторій повинен містити в корені текстовий файл ***README.md*** або ***README.rst***, у якому присутній наступний вміст:
    - Короткий опис застосунку, що він робить (напр. Quadratic Equation Solver)
    - Розрахунок номеру варіанту та опис варіанту
    - Інструкція, як зібрати проект та запустити тести
    - посилання на коміт, на якому впали тести на CI