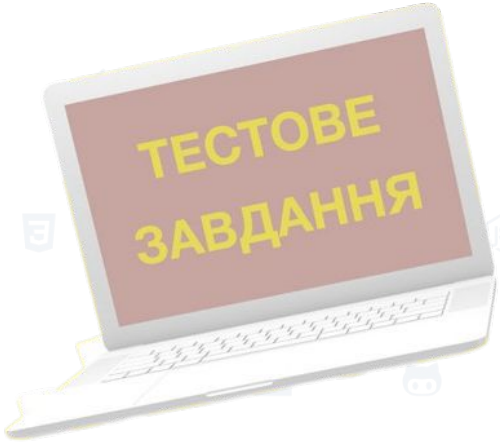
Lesson 26

01.03.2024



Джун вперше бачить тестове завдання.

Тестові завдання в компанії можуть бути однакові для різних рівнів розробників.

Які часті теми тестових завдань?

- алгоритми
- парсинг
- веб-сервіси (включаючи ботів)
- комбінації цих тем

Як читати тестове завдання?

- прочитайте кілька разів
- виключіть все "бажано", якщо вам ці технології незнайомі
- якщо невідомі технології залишилися, прочитайте, що це таке. можливо, ви знайдете їй відому заміну.
- визначте вхідні та вихідні дані.
- при можливості поставте уточнюючі питання

Що має бути у тестовому?

- воно має працювати (хоч його і не запускатимуть)
- вся програма не в одному методі main()
- якщо є вимоги щодо формату вхідних/вихідних даних - суворо дотримуйтесь
- єдиний стиль коду / чистота коду
- неймінг
- збиратися maven/gradle (для підключення залежностей)
- unit тести
- логування
- обробка винятків
- readme (як запустити, параметри, що робить додаток, особливості)
- git (показати порядок та хід розробки)

Буде чудово, якщо:

- якщо це веб-сервіс - розмістіть, щоб роботодавець сам помацав
- хай додаток буде універсальним (зовнішня конфігурація)
- інтеграційне тестування
- буде демонстрація роботи (1-3 хв)
- Docker
- git (будь-яке виконане тестове = +1 проект у вашому репозиторії)

Рекомендації

- використовуйте сторонні бібліотеки. Уникайте низькорівневих операцій.
- уникайте ранніх оптимізацій (можете зайнятися після розв'язання задачі)
- використовуйте ООП (одне завдання – один клас)
- тримайте під рукою список популярних алгоритмів та патернів
- використовуйте силу існуючих структур даних
- якщо вам здається завдання неадекватне, обговоріть із спільнотою та колегами