Úkol 10

Fibonacciho posloupnosti

Tento projekt ukazuje **tři způsoby výpočtu n-tého prvku Fibonacciho posloupnosti**, indexované od 0:

1. calcNerek(int n) – **nerekurzivní přístup** (iterativní řešení).
2. calcRek(int n) – **rekurzivní přístup** (pomalejší, ukazuje princip).
3. calcRekTable(int n) – **dynamické programování** s použitím **tabulky (memoizace)**.

Proč tři různé metody?

1. Iterativní (calcNerek)

- Efektivní a rychlý způsob bez rekurze.
- Paměťově úsporný – používá jen dvě proměnné.

2. Rekurzivní (calcRek)

- Ukazuje princip, ale je **neefektivní pro větší n** (exponenciální časová složitost).
- Nepoužívá mezipaměť, dochází k opakovanému výpočtu stejných hodnot.

3. Dynamické programování (calcRekTable)

- **Efektivní kombinace rekurze a mezipaměti (tabulky)**.
- Tabulka ukládá mezivýsledky – **zabraňuje zbytečným výpočtům**.
- Výstup ukazuje stav tabulky v jednotlivých krocích, což usnadňuje sledování výpočtu.

Co dělá init()?

Metoda init(int n) inicializuje pole table, které slouží jako paměť pro již vypočítané hodnoty Fibonacciho čísel.

- Pole má velikost n + 1.
- Hodnoty jsou na začátku nastaveny na -1, což značí, že ještě nebyly vypočítány.

Výhody použití tabulky

- Zrychlení výpočtu – místo exponenciálního času (O(2^n)) se dostáváme na **lineární čas** O(n).
- Zabránění opakovaným výpočtům.

- Přehledný záznam průběhu výpočtu.

Metody indexOf() v Java ArrayList

Soubor IndexOfTest.java

Obsahuje pět testovacích případů:

1. **První výskyt prvku**: Ověření, že metoda vrací index prvního výskytu zadaného prvku.
2. **Prvek neexistuje**: Ověření, že metoda vrací -1, pokud prvek není v seznamu.
3. **Prázdný seznam**: Ověření, že metoda vrací -1 pro prázdný seznam.
4. **Prvek je null**: Ověření, že metoda správně identifikuje index prvku null.
5. **Více výskytů prvku**: Ověření, že metoda vrací index prvního výskytu i při více výskytech.

Porovnání řetězců v Javě

Implementovat vlastní metodu pro lexikografické porovnání dvou řetězců bez použití vestavěných metod jako equals() nebo compareTo().

Soubor StringUtils.java

Obsahuje metodu customCompare, která:

- Porovnává dva řetězce znak po znaku.
- Vrací:
    - 0 pokud jsou řetězce stejné.
    - -1 pokud je první řetězec lexikograficky menší.
    - 1 pokud je první řetězec lexikograficky větší.

Testování

Soubor StringUtilsTest.java obsahuje metodu main, která testuje customCompare na různých dvojicích řetězců, včetně:

- "ABCDEF" vs "ABCD" → očekáváno 1
- "ABCD" vs "ABCDEF" → očekáváno -1
- "ABCD" vs "ABCD" → očekáváno 0
- "ABCD" vs "ABCE" → očekáváno -1
- "ABCE" vs "ABCD" → očekáváno 1