Exercitiu - Obtinerea detaliilor numarului

Prezentare generala

Implementeaza o clasa, numita LazyNumberDetails, a caror obiecte stocheaza fiecare un numar natural si ofera utilizatorului (obiectului respectiv) urmatoarele detalii despre numarul stocat:

1. Este un [numar prim](https://edu.devmind.ro/java-1/algoritmi-de-baza-i#numere_prime_divizori_descompunere)?
2. Este un [numar perfect](https://edu.devmind.ro/java-1/algoritmi-de-baza-i#numere_perfecte)?
3. Este un [numar magic](https://edu.devmind.ro/java-1/algoritmi-de-baza-i#numere_magice)?

Toate cele 3 operatii implementate de clasa sunt considerate a fi **operatii intens computationale**. Prin urmare, acestea trebuie efectuate de un numar cat mai mic de ori, doar atunci cand este strict necesar. Devenind treptat un programator din ce in ce mai iscusit, pasionat si determinat, sarcina ta este sa realizezi o implementare **eficienta** a clasei LazyNumberDetails, care integreaza cele doua mecanisme puternice de optimizare: [Lazy evaluation](https://edu.devmind.ro/java-1/oop-i/exercitii-oop-i-optimizare#lazy_evaluation) si [Caching](https://edu.devmind.ro/java-1/oop-i/exercitii-oop-i-optimizare#caching).

Concret, clasa LazyNumberDetails va contine cel putin urmatoarele campuri si metode:

[**LazyNumberDetails.java**](https://edu.devmind.ro/_export/code/java-1/oop-i/exercitii-oop-i-optimizare?codeblock=0)

public class LazyNumberDetails {

private int number;

public LazyNumberDetails(int number) {...}

public void updateNumber(int number) {...}

public boolean isPrime() {...}

public boolean isPerfect() {...}

public boolean isMagic() {...}

}

Metodele publice prezentate in schita anterioara sunt doar cele care vor fi apelate de utilizator. In cadrul implementarii, va trebui sa definesti orice alte metode si campuri necesare pentru a respecta functionalitatea ceruta, detaliata in continuare.

Functionalitate

Un obiect de tip LazyNumberDetails va contine intotdeauna un numar natural (i.e. campul number) despre care va raspunde, prin intermediul metodelor publice, la intrebarile aferente (i.e. isPrime?, isPerfect?, isMagic?).

Numarul respectiv este primit (obligatoriu) la construirea obiectului, iar acesta poate fi schimbat (oricand) pe parcursul programului de catre utilizatorul obiectului, folosind metoda updateNumber(int). Din momentul in care numarul intern a fost schimbat, toate metodele de interogare (i.e. isPrime?, isPerfect?, isMagic?) vor returna rezultate aferente noii valori. Cu alte cuvinte, metodele de interogare vor afisa mereu informatii actualizate, despre **ultimul / cel mai recent numar stocat**.

Optimalitate

Deoarece toate cele 3 interogari (i.e. isPrime?, isPerfect?, isMagic?) sunt considerate operatii costisitoare, acestea trebuie executate de un numar minim de ori. Acest lucru va fi asigurat prin implementarea celor doua **modele de optimizare**:

* Modelul [Lazy evaluation](https://edu.devmind.ro/java-1/oop-i/exercitii-oop-i-optimizare#lazy_evaluation) asigura faptul ca oricare algoritm intens computational este executat doar daca / atunci cand utlizatorul solicita rezultatul.
* Mecanismul de [Caching](https://edu.devmind.ro/java-1/oop-i/exercitii-oop-i-optimizare#caching) realizeaza stocarea rezultatului pentru oricare algoritm intens computational (i.e. dupa executia acestuia). Astfel, rezultatul memorat este disponibil pe viitor, daca utilizatorul efectueaza din nou aceeasi interogare, si poate fi returnat **direct** (i.e. fara a executa din nou algoritmului in cauza).

Exemplu de rulare

Sa consideram un scenariu concret de rulare, pentru a intelege mai bine functionarea obiectelor clasei:

LazyNumberDetails lnd;

lnd = new LazyNumberDetails(23); *// (1) \*\*none (lazy eval.)\*\* -> no user action*

*// User instruction | What happens? | Explanation*

lnd.updateNumber(10); *// (2) \*\*none (lazy eval.)\*\* -> no user action*

lnd.updateNumber(17); *// (3) \*\*none (lazy eval.)\*\* -> no user action*

lnd.isPrime(); *// (4) !!intensiveComputation!! -> prime algorithm for 17*

lnd.isPrime(); *// (5) \*\*cached\*\* -> saved result from prime alg*

lnd.updateNumber(28); *// (6) \*\*none (lazy eval.)\*\* -> no user action*

lnd.isPerfect(); *// (7) !!intensiveComputation!! -> perfect algorithm for 28*

lnd.isPrime(); *// (8) !!intensiveComputation!! -> prime algorithm for 28*

lnd.isPerfect(); *// (9) \*\*cached\*\* -> saved result from perfect alg*

lnd.isMagic(); *// (10) !!intensiveComputation!! -> magic algorithm for 28*

lnd.isPrime(); *// (11) \*\*cached\*\* -> saved result from prime alg*

lnd.updateNumber(12); *// (12) \*\*none (lazy eval.)\*\* -> no user action*

lnd.isMagic(); *// (13) !!intensiveComputation!! -> magic algorithm for 12*

lnd.isMagic(); *// (14) \*\*cached\*\* -> saved result from magic alg*

...

Remarcam ca fiecare operatie costisitoare (i.e. executia algoritmului prime, perfect sau magic) se va realiza practic doar intr-un singur caz:

1. Daca utilizatorul realizeaza (explicit) o interogare de forma isX(), iar acea interogare nu a mai fost efectuata pana atunci, pentru numarul stocat.
   * Instructiunile corespunzatoare: (4), (7), (8), (10) respectiv (13).

In toate celelalte cazuri, nu se va executa nicio operatie costisitoare (i.e. aferenta algoritmului prime, perfect sau magic):

1. Daca utilizatorul a mai realizat in trecut interogarea isX() pentru numarul stocat, atunci rezultatul este deja stiut. Prin urmare, acesta va fi returnat direct.
   * Hint: Cum poate fi deja stiut? - La executia initiala a interogarii isX(), atunci cand a fost necesara executia algoritmului X (intens computational), se stocheaza (i.e. intr-un alt camp al obiectului) rezultatul calculat de algoritm. Ulterior, acel rezultat va fi returnat direct, fara a mai rula algoritmul.
     + Acesta este mecanismul de [Caching](https://edu.devmind.ro/java-1/oop-i/exercitii-oop-i-optimizare#caching) ;).
   * Instructiunile corespunzatoare: (5), (9), (11) si (14).
2. Daca utilizatorul nu realizeaza niciodata interogarea isX() pentru numarul curent (number), atunci niciun algoritm intens computational nu a fost executat (inutil).
   * Acesta este mecanismul de [Lazy evaluation](https://edu.devmind.ro/java-1/oop-i/exercitii-oop-i-optimizare#lazy_evaluation) ;).
   * Instructiunile corespunzatoare: (1), (2), (3), (6) si (12).