Chirila Alexandru
324CB

Comenzile implementate

set perioada <nr_milisecunde>: Seteaza perioada minima de asteptare intre preluarea a 2 comenzi (folositor la rulare cu redirectare de input dintr-un fisier deoarece, cel putin pe Windows, este posibila terminarea mai rapida a unei comenzi decat a alteia care va fi fost data mai tarziu datorita multithreading)

print administratori: Afiseaza lista de administratori

print brokeri: Afiseaza lista de brokeri

print clienti: Afiseaza lista de clienti

print produse: Afiseaza lista de prduse

client <id_client>: afiseaza clientul cu id-ul respectiv

client <id_client> print_produse: Afiseaza lista de prduse, dar cu comision aplicat pentru pretul minim in functie de tipul clientului (persoana fizica sau juridica) si de numarul de licitatii castigate

client <id_client> licitez <id_produs> <suma_maxima>: Clientul intra intr-o licitatie (sau creeaza una noua) pentru produsul cu id-ul specificat DACA suma maxima declarata este mai mare sau egala decat pretul minim al produsului cu comision aplicat (pentru ca altfel oricum nu ar putea cumpara produsul chiar daca ar castiga licitatia)

client <id_client> propun <id_produs> <suma>: Clientul propune o suma pentru produsul respectiv. Suma trebuie sa fie mai mica sau egala decat suma maxima declarata initial cu comisionul extras (sau suma propusa cu comision aplicat trebuie sa fie mai mare sau egala decat suma maxima declarata initial). Daca clientul deja a facut o propunere la pasul curent din licitatie, atunci aceasta propunere este ignorata

client <id_clinet> skip <id_produs>: Cu aceasta comanda clientul cere actualizarea propunerii lui cu suma propusa anterior (sau 0 daca inca nu a propus nicio suma). Daca propunerea clientului a fost deja actualizata la pasul curent al licitatiei aceasta comanda este ignorata

stop: Termina executia aplicatiei.

Chirila Alexandru
324CB

client <id_client> quit <id_produs>: Cu aceasta comanda clientul poate parasi licitatia

print casadelicitatii: afiseaza toate listele memorate de casa de licitatii (administratorii, brokerii, licitatiile, clientii si produsele)

new angajat administrator: Creeaza un nou administrator

new angajat broker: Creeaza un nou broker

new client psf <nume> <adresa> <data_de_nastere_valida_format_zz.ll.aaaa>: Creeaza un client persoana fizica, cu numele <nume>, adresa <adresa> si data de nastere zz.ll.aaaa

new client psj <nume> <adresa> <tip_companie> <capital_social>: Creeaza un client persoana juridica, cu numele <nume>, adresa <adresa> si capitalul social <capital_social>

new produs tablou <nume> <nume_pictor> <culori> <an> <pret_minim>: Creeaza un tablou cu pretul minim <pret_minim>, numele <nume> pictat de pictorul <nume_pictor> in culori <culori> in anul <an>

new produs mobila <nume> <tip_mobila> <material> <an> <pret_minim>: Creeaza un obiect de mobila cu numele <nume> de tipul <tip_mobila> cu pretul minim si facut din materialul <material> in anul <an>

new produs bijuterie <nume> <material> <piatra_pretioasa> <an> <pret_minim>: Creeaza o bijuterie cu numele <nume> facut in anul <an> din materialul <material> si cu pretul minim <pret_minim>. <piatra_pretioasa> poate fi true sau false si descrie daca bijuteria are sau este o piatra pretioasa

Chirila Alexandru
324CB

Despre implementare

Am folosit multithreading la createa de administratori, brokeri, clienti, produse si la cerintele clientilor pentru licitatii. Threadul principal se ocupa de comenzile de afisare, deci clientii pot vizualiza lista de produse in timp ce un administrator sau mai multi adauga un produs.

Design pattern-uri folosite (in forme mai mult sau mai putin alterate):

- **Observer**: Clasa Broker are o variabila statica noutati (la extinderea la mai multe case de licitatie, aceasta variabila statica devine un ArrayList, fiecare element pentru cate o casa de licitatii)
- **Command**: In loc sa folosesc o multime de clase am folosit metode statice, clase care extind Thread si care creeaza instante pe care le adauga in lista corespunzatoare a casei de licitatii: MkAngajat, MkProdus, MkClient, MkLicitatie
- **Singleton**: Casa de licitatii are o singura instanta initializata in context static (la extinderea la mai multe case de licitatii aceasta variabila devine un MyArrayList – clasa mentionata in JavaDoc)
- **Builder**: Setteri in stil Builder pentru clasele Produs, PersoanaFizica, PersoanaJuridica, Bijuterie, Tablou si Mobila
- **Facade**: Lista de LicCli (clientul, licitatia in care participa clientul si suma maxima declarata de client) memorata de fiecare broker si lista de LicBrok (brokerul asignat clientului si licitatia in care participa clientul) memorata de fiecare client.
- **Decorator**: Clasa MyArrayList<E> ce extinde ArrayList<E> si are un camp Lock, o metoda mai eleganta toString, metoda getRandomElem ce returneaza un element la intamplare din lista si metoda getElem ce returneaza obiectul din lista egal cu cel primit ca parametru sau null. ListaProduse, de asemenea, are o metoda custom toString care afiseaza produsele cu pretul minim cu comisionul aplicat al clientului care solicita vizualizarea produselor.

Chirila Alexandru
324CB

Mod de primire al comenzilor (structura arborescenta de cazuri):

```
new
        angajat
                administrator
                broker
        produs
                bijuterie
                tablou
                mobila
        client
                psf <nume> <adresa> <data_de_nastere_valida_format_zz.ll.aaaa>
                psj <nume> <adresa> <tip_companie> <capital_social>
set
        perioada <nr_milisecunde>
print
        clienti
        produse
        brokeri
        administratori
        licitatii
        casadelicitatii
client <id_client>
        print_produse
        licitez <id_produs> <suma>
        propun <id_produs> <suma>
        skip <id_produs>
        quit <id_produs>
set
        perioada <nr_milisecunde>
stop
```

Chirila Alexandru
324CB

Dependente:

Ce are voie un client sa stie? Licitatiile in care participa si brokerul mediator al acestuia pentru fiecare licitatie => Lista de LicBrok

Ce are nevoie un broker sa stie? Clientii si licitatiile in care participa acestia => Lista de LicCli

Ce este necesar sa se retina intr-o licitatie? Clientii participanti si suma propusa de fiecare dintre acestia => Lista de obiecte Propunere

Transmiterea comenzilor de la clienti catre Licitatii:

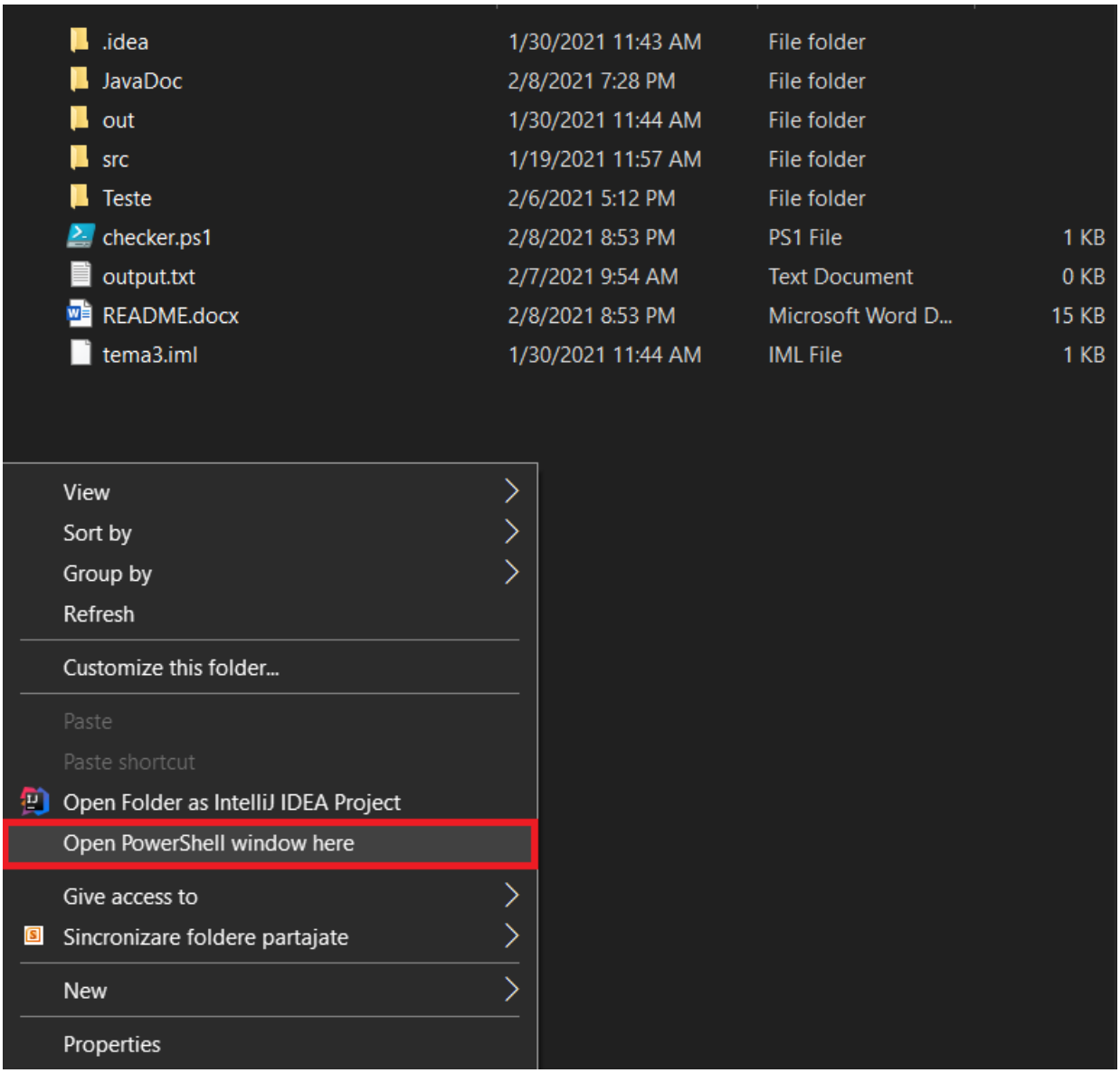
Broker -> CasaDeLicitatii -> Licitatii

Mai multe detalii de implementare in JavaDoc si in fisierele sursa (.java).

Chirila Alexandru
324CB

Rularea checkerului pentru Windows 10

Executati fisierul “checker.ps1” (fisier de tip PowerShell script). Este posibil ca PowerShell sa nu vrea se execute scripturi fara a schimba anumite setari in computer. In acest caz dati click dreapta in interiorul folderului “tema3” (cel in care se afla checkerul) in timp ce tineti apasata tasta SHIFT si deschideti o fereastra PowerShell.



Deschideti fisierul checker.ps1 cu un editor de text si copiati continutul acestuia in fereastra de PowerShell pe care ati deschis-o. Apasati enter pentru a rula.

Chirila Alexandru
324CB

Daca nu se afiseaza secvente lungi, atunci totul este ok. Altfel difera fisierele .out de fisierele .ref.

Rulare checker pe Linux

Deschideti terminalul, intrati in folderul tema3 si executati fisierul checker.sh.

Daca nu se afiseaza secvente lungi, atunci totul este ok. Altfel difera fisierele .out de fisierele .ref.

Mentionez ca datorita multithreading este posibila obtinerea a 2 output-uri diferite la rularea proiectului de doua ori pe acelasi input, mai ales la rularea intr-o masina virtuala.

Scopul testelor

Testul 0: Cerintele standard

Testul 1: 2 licitatii care se desfasoara simultan

Testul 2: 2 licitatii se desfasoara simultan si un client participa la ambele

Testele 3-9: Umplutura (toate functionalitatile cheie au fost testate in cadrul testelor 0, 1 si 2).