ServerAplication linia 10 pornesc un socket la portul 3500 care asteapta conexiuni. De fiecare data cand se conecteaza cineva iese din functia accept se porneste un thread pentru client (linia 13)

Merg in ClientThread unde avem un canal de comunicare linia 105-108(asemanator cu console-in console out). La linia 112 un thread pe care intra prima data programul cand dau pe search for opponent. El trateaza username-ul si care este requestul curent. Dupa clientul va tot genera cereri de refresh. Fiecare request trimis de la client este tratat o data si atat. E mai simplu decat sa avem conexiunea deschisa mereu.

Ce se intampla important este in treatRequest (linia 357). Avem 3 lucruri importante. IntParams (vedem ce ne-a trimis clientul) si outparams(punem ce vrem sa trimita la client).

Rezultatele si requesturile le putem vedea in Codes. De exemplu la linia 5 serverul poate spune ca (totul e bine) sau linia 7(am intampinat o eroare) sau (username-ul asta deja este luat)...

(linia 360) Revenim la ClientThread in outparams unde vom pune un UserId pe care il asociem numelui lui. Acel rezultat il dam ca si user id(la linia 75) este o functie registerUsername unde serverul face o asociere cu ajutorul id-ului primit folosindu-se de usrAssoc(linia 77). Daca deja am numele useru-lui inseamna ca acel user deja este conectat si joaca un joc deci (linia 78) UsernameTaken(). (linia 83) Daca se da un userId care deja este luat se mai reface randomul de un numar limitat de ori. (Linia 91) daca totul este bine la userID ii punem numele in sine.

De exempl vom avea (3367 - Andrei si 4125 - Iustin). (Asociem un numar cu un nume). Din punctul asta clienti vor folosi numele pe care il vor salva si il vor folosi in joc.

Mergem inapoi la linia 360. Daca e totul bine va da serverSuccess; Daca nu a gasit loc pentru el usernameul inseamna ca este luat. Cand clientul se deconecteaza dam unregisterUsername. Cam in asta consta conectarea si deconectarea

De la linia 376 avem 4 chestii importante. Cand suntem in cautare de oponent, cand esti in joc si dai refresh ca sa vezi daca este tura ta, cand dai cu zarul si cand faci mutarea. Toate astea trebuie trimise catre server. Fiecare din ele are o functie separata. Cel mai simplu este cand cautam oponent. Acea functie se gaseste la linia 148 treatMatchmaking. Cand cautam oponent se obtine user id-ul de la client. Incepe sa se caute daca nu cumva exista un joc deja pregatit pentru el si el nu stie asta(se cauta prin toate jocurile si se uita dupa Id). Salvam forThis ca sa mai trimitem informatiile inapoi daca este in jocul asta. L alinia 175 spune ca daca amandoi au fost notificati (sunt gata sa joace) li se trimite si tabla(nr de piese s piesele de pe fiecare). In cazul contrar se initializeaza tabla. Daca tabla nu este initializata trebuie initializata (185 pana la 226). In final daca ambii jucatori au fost notificati atunci ei nu mai asteapta(ii scoate din asteptare). Linia 239 => daca nu exista jucator in linia de asteptare se va pune jucatorul intr-o linie de asteptare

La lini 240 vede daca sunt mai mult de doi useri. Daca sunt atunci se scoate userul care vrea sa joace si mai cautam un alt user in toti useri. Mai exact scot id-ul curent si mai caut altul. La linia 251 clientul inca asteapta un alt user in cazul in care acesta nu a fost gasit.

Functia treatInGame este cea mai simpla functie. Ea spune in felul urmator: Daca nu este tura mea atunci i se trimite inapoi (Asteapta deoarece momentan joaca inamicul si zarurile pe care acesta le-a dat tura trecuta). Am observat ca treaba asta merge cand vrea ea (nu stiu de ce). Acum suntem in punctul in care este tura mea. Pot primi informatii din tura inamicului si sa dau inapoi nr de piese si care jucatori detin ce piese. Prin g-urile din ghilimele de la 278 si 279 se dau inapoi ce piese au fost scoase prin eliminare(de exemplu daca albul a loat o piesa de la negru si o pune la mijlocul tablei). Urmatorul if vede daca inca nu am ales cine da cu zarul iar dupa el se vede daca jocul este in rulare si este tura mea.

Functia de dat cu zarul (treatRoll lnia 393) este putin mai ciudata pentru ca trebuie sa vedem urmatoarele lucruri: meciul este la perioada la care trebuie sa se aleaga cine incepe meciul si inca nu a dat nimeni cu zarul sau meciul este la perioada in care nu s-a ales cine sa dea primul si a dat cineva cu zarul sau meciul este la perioada in care nu s-a ales cine sa dea primul si amandoi au dat cu zarul (si trebuie vazut care a dat mai mare sau daca trebuie dat din nou fiindca a fost egal) sau daca suntem in timpul jocului. Dupa ce trece prin aceasta functie va trimite clientul catre functia clientMoved linia 299.

La facut mutarea se codifica mutare intr-un anumit mod. Folosim for pentru a pune un numar de maxim 4 mutari primite de la un player. O mutare poate avea 4 efecte(cand luam o piesa a inamicului marat g ce este pusa separat deoarece pisele se duc in mormant si nu mai trebuie retinut locul de pe care sa intamplat mutarea)(in cazul celorlalte tipuri de mutari trebuie sa stim si patratica de pe tabla de unde a fost facuta acea mutare. Codificarea este facuta ceva in genul f19t22r22 (from 19 to 22 remove 22, remove 22 este optionala). La final se va schimba randul (game.turn)

In mare parte serverul doar tine evidenta lucrurilor precizate mai sus si da mai departe prin niste obiecte ce sunt declarate la inceputul programului (incep de la linia 18)

In interfata RequestCallBack avem functia callback unde primim un code si un raspuns.

In clasa Net: avem urmatoarele :

* requesturile active ale clientului curent
* functia makeRequest()- care primeste un request, un request id si functia callback care va fi apelata. Clasa request este un thread in care facem conexiunea cu serverul, trimitem informatiile si citim raspunsul de la server. Tot aici vom decodifica parametrii, daca exista. Daca primim raspuns vom apela functia de callback cu Code-ul si parametrii corespunzatori.

In GameApplication se realizeaza fereastra de joc. Aceasta are doua posibilitati MenuPanel – fereastra pentru conectare sau fereastra de joc propriu-zisa GamePanel. Aici avem doua functii switchToMenu() si switchToGame(). Tot aici avem si un constructor in care se va deschide fereastra cu parametrii corespunzatori: titlul, marimea. Daca se inchide fereastra se va face un request de disconnect catre server. Daca primim de la server raspuns ca am fost asignati unui joc se va face switchToGame.

In clasa MenuPanel avem urmatoarele:

* Functia *onConnect():* cand apasam din client “Search for opponent” se realizeaza un request la adresa si portul specificate, si cu parametrii dati in cazul acesta “username-ul”, atunci cand primim raspuns de la server se va apela functia lambda (un callback). Daca clientul s-a conectat, primim un id asociat numelui nostru curent, altfel daca usernameul nu e disponibil afisam un mesaj.
* Parametrii urmatori: titlu, doua elemente JLabel si doua elemente JTextField si butonul pentru search.
* Un constructor in care asezam elementele in fereastra.
* Functia de connectSucces(): apelata din functia onConnect() daca avem un username valid. Aici vom sta in asteptare si vom face requesturi de refresh catre server pentru a afla daca ne-a fost asignat un adversar- lucru observat in variabila foundGame. Tot aici avem un thread in care se va realiza requestul catre server, pentru a afla daca ne-a fost gasit un meci. Daca se va gasi un meci se va apela functia gameFound();
* Functia gameFound(): care va apela switchToGame, se va schimba fereastra de conectare in fereastra de joc.

In GamePanel avem:

* functia reinitialise(): din raspunsul primit de la server salvam pozitia pieselor. In board salvam in doua map-uri: cate piese sunt la ce pozitie si ce piese sunt la ce pozitie. – getPieceCountMap si getPieceTypeMap
* campuri pentru culori
* campuri pentru pozitiile de pe tabla: partea de tabla din stanga, respectiv din dreapta.
* Un Board in care salvam elementele de pe tabla.
* Un JLabel in care se va afisa textul pe tabla.
* Funtia painComponent care va apela drawBoard() – cu ajutorul acesteia vom desena tabla. De aici se va apela drawPieces si drawDice si daca este cazul, drawGraveyard si in mijlocul tabelei mesajul de informatie.
* Functia startCom() – inceputul comunicarii, aici avem un thread care ruleaza la infinit, in care se face un request la fiecare 200 milisecunde, unde cer statusul jocului. Si aici putem primi serverUpdateBoard- a fost mutata o piesa, dupa mutarea inamicului, serverNotYourTurn- nu este tura ta, astepti sau serverYourturn- tura noastra, atunci vom da enable pentru a da cu zarul.