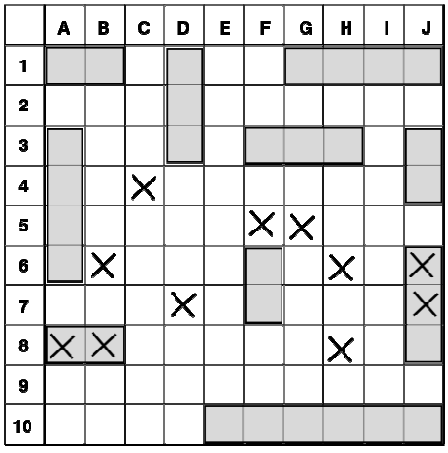
Univerzitet u Beogradu
Elektrotehnički fakultet
Odsek za SI

Praktikum iz objektno orijentisanog programiranja
13S112POOP
2015-2016

Treći projektni zadatak (Java)

Napisati pakete klasa na programskom jeziku Java sa odgovarajućim atributima, metodama i konstruktorima kojima se realizuje igra "Pomorska bitka", za više igrača (https://en.wikipedia.org/wiki/Battleship_(game)). *Tradicionalno*, radi se o igri poteznog karaktera za 2 igrača. Dimenzije table za igru svakog igrača su 10×10 polja. U takvoj tabli, igrači najpre moraju da rasporede (sakriju) bojne brodove svoje flote na pozicije nepoznate drugom igraču. Nakon raspoređivanja, brodovi su nepokretni. Svaki brod zauzima jedno ili više uzastopno povezanih polja table, horizontalno ili vertikalno. Između dva broda mora postojati najmanje jedno polje rastojanja horizontalno, vertikalno ili dijagonalno. Dimenzije brodova su 1, 2, 3 ili 4 polja, a broj odgovarajućih brodova u floti je 4, 3, 2 i 1, respektivno. U nastavku teksta, *segment* broda će označavati jedno polje koje deo datog broda zauzima. Igrači naizmenično navode koordinate polja u tabli za koje pretpostavljaju da bi mogle da budu lokacija jednog segmenta protivničkog igrača. Navođenje koordinata se tumači kao gađanje datog polja. Protivnički igrač (koji je bio gađan) dužan je da odgovori "pogodak" ili "promašaj". Svaki igrač je dužan da prati polja na koja je gađao, da ne bi više puta gađao u isto polje. Pogođen segment broda smatra se uništenim (neoperativnim), a brod čiji su svi segmenti neoperativni smatra se potopljenim. Pobeđuje onaj igrač koji prvi potopi sve protivnikove brodove.

Ilustracija table za igru u tradicionalnoj realizaciji prikazana je na slici 1. U trećem projektnom zadatku se ne očekuje pravljenje grafičkog interfejsa, ali se očekuje ispisivanje sadržaja tabli u tekstualnom obliku. U tom smislu, ovu sliku ponajpre treba koristiti kao ilustraciju u cilju razumevanja očekivanog ponašanja nekih klasa. U trećem projektnom zadatku, neki elementi igre će odudarati od tradicionalnih parametara igre.



Slika 1. Ilustracija izgleda table dimenzija 10×10 polja. Sivom su označeni sakriveni brodovi (nije poštovan tradicionalni broj i veličina brodova), a sa X su označene lokacije koje je gađao protivnički igrač.

Realizacija igre se sastoji iz dve komponente: serverske i klijentske. Serverska komponenta predstavlja nadzornika igre, koja sadrži table za igru svih igrača i postavlja pravila igre. Klijentske komponente predstavljaju igrače, koji sadrže tablu gde dati igrač raspoređuje borbena plovila u svoje flote i table na kojima igrač beleži svoja gađanja brodova drugih igrača (za svakog protivničkog igrača po jedna tabla). Klijenti se prijavljuju serveru, uz eventualni unos lozinke, ako igra nije javnog karaktera. Po početku igre, server prijavljenim klijentima dojavljuje dimenzije tabli za igru, karakteristike (broj i veličinu) plovila sa kojima se igra i trajanje jednog poteza. Igrači na raspolaganju imaju 60 sekundi da rasporede svoja plovila po tabli. Neraspoređena plovila ne učestvuju u sastavu flote igrača. Nakon toga, igrači imaju 30 sekundi (vreme poteza) da zadaju koordinate pozicija na koje žele da gađaju u tablama drugih igrača. U jednom potezu igrač može da izda onoliko naredbi gađanja koliko ima operativnih segmenata svojih brodova. Nakon formiranja spiska naredbi gađanja, igrač izdaje naredbu za paljbu kojom se serveru dojavljuje spisak koordinata. Naredbu gađanja je moguće izdati samo tokom vremena predviđenog za jedan potez. Po isteku datog vremena server će zanemariti naredbu gađanja. Server zatim svim klijentima javlja sva gađana polja, a klijentima koji su izgubili sve brodove javlja da je njihova igra završena. Pri tom, igrači čija je igra završena mogu da posmatraju ostatak bitke. Igrač koji jedini preostane sa operativnim brodovima pobeđuje. Ako takvog igrača ne bude posle poslednjeg gađanja, igra je nerešena.

Klijentska i serverska komponenta treba da budu realizovane kao nezavisne izvršne komponente koje međusobno (klijenti sa serverom) komuniciraju mrežnim putem.

Napomena: dopune protokola označene su zelenom bojom, a komande žutom pozadinom.

1. Protokol komunikacije

Protokol komunikacije između klijenta i servera je sledeći:

1. obeležavanje koordinata
   1.1. koordinate u tabli se obeležavaju četvorocifrenim celim decimalnim brojevima. Dve cifre najveće težine označavaju vrstu, a preostale dve cifre kolonu u tabli. Tako koordinate 0407 označavaju 4. vrstu i 7. kolonu. Pretpostavlja se da dimenzije table za igru ne mogu biti preko 100 vrsta odnosno kolona.
   1.2. tabla se obeležava oznakom {*x*}, gde *x* predstavlja ime kojim se igrač predstavi igri.

2. igrač (klijent) pristupa igri (serveru):
   2.1. slanjem tekst poruke JOIN *name*, gde je *name* znakovni niz koji predstavlja izabrano ime igrača
      2.1.1. ako je server sa ograničenim pristupom, server odgovara na zahtev slanjem tekst poruke PASSWORD_REQUIRED
      2.1.2. ako je server javno dostupan, server prihvata igrača i odgovara na zahtev slanjem tekst poruke WELCOME *id*, gde *id* predstavlja jedinstven numerički identifikator igrača koji server generiše
   2.2. slanjem tekst poruke JOIN *name /password*, gde *password* predstavlja tekst proizvoljne dužine koji predstavlja lozinku za pristup serveru
      2.2.1. ako je server sa ograničenim pristupom, a lozinka nije ispravna, server odgovara na zahtev slanjem tekst poruke ACCESS_DENIED

2.2.2. ako je server javno dostupan ili ako je lozinka ispravna, server prihvata igrača i odgovara na zahtev slanjem tekst poruke WELCOME *id*, gde *id* predstavlja jedinstven numerički identifikator igrača koji server generiše

2.3. ako je ime igrača već iskorišćeno, server odgovara porukom DUPLICATE_NAME

2.4. server odbija sve poruke koje zahtevaju identifikaciju igrača ako se ne dostavi postojeći (dodeljen) identifikacioni broj

3. igrač odustaje od dalje igre slanjem tekst poruke QUIT *id*
   3.1. server odgovara na poruku slanjem tekst poruke BYE; igrač ne mora da primi odgovor.

4. server može da obavesti igrača da je udaljen iz igre slanjem tekst poruke FORCE_DISCONNECT

5. igrač od igre može da zatraži trenutno stanje igre slanjem tekst poruke STATE
   5.1. igra na upit odgovara tekst porukom:
      5.1.1. WFP *x*/*y* (waiting for players), što označava da igra čeka da se priključi još igrača; *x* predstavlja broj trenutno priključenih igrača, a *y* maksimalan broj igrača
      5.1.2. DS *time* (deploy ships), što označava da igra čeka još *time* sekundi da igrači rasporede svoje brodove
      5.1.3. R *x time* (round), što označava da je u toku potez rednog broja *x* i da igra čeka još *time* sekundi da igrači dojave listu svojih gađanja
      5.1.4. U (update), što označava da igra računa pogotke i javlja igračima ishod

6. po ulasku u stanje *deploy ships*, igra obaveštava sve igrače koji su pristupili igri porukom D=*size*;S(*x*1)=*y*1;...;S(*x*n)=*y*n, gde je *size* ceo broj i predstavlja veličinu table za raspoređivanje brodova (broj vrsta i kolona), *x* predstavlja broj segmenata broda, a *y* količinu brodova. Na primer S(3)=5 označava da ima 5 brodova veličine 3 segmenta.
   6.1. igrači odgovaraju vraćanjem primljene poruke: CONFIRM_DEPLOY *id primljena_poruka*
      6.1.1. svim igračima koji se nisu odazvali, server ponavlja poruku određen broj puta. Igrači koji se ni nakon toga ne odazovu bivaju izbačeni iz igre. Kada su svi preostali igrači u igri potvrdili da su primili poruku, počinje merenje vremena predviđenog za raspoređivanje jedinica
7. igrač može da igri pošalje poruku kojom javlja raspored svojih brodova:
   7.1. SHIP_LAYOUT *id* S(*x1*)=[*C1*,*C2*,...,*Ck*];S(*x2*)=[*C1*,*C2*,...,*Cl*];...;S(*xn*)=[*C1*,*C2*,...,*Cj*], gde *id* označava jedinstveni identifikator igrača, *x* označava broj segmenata broda, a *C* označava koordinate polja na kojima su smešteni segmenti broda. Koordinate su izražene na način opisan u tački 1. Koordinate treba da budu uređene rastuće po dimenziji po kojoj se menjaju. Na primer, ako je brod od 3 segmenta smešten horizontalno u vrsti 4, od kolone 5 do kolone 7, vrednosti C1,C2,C3 treba da budu 0405, 0406, 0407, respektivno.
      7.1.1. igra odgovara porukom LAYOUT_ACCEPTED ako je igra u stanju *deploy ships* i zadati raspored ne narušava pravila igre i ako igrač nije rasporedio više brodova nego što je dozvoljeno.
      7.1.2. igra odgovara porukom LAYOUT_REJECTED ako zadati raspored nije prihvatljiv ili ako igra nije u stanju *deploy ships* ili ako je za datog igrača prethodno već prihvaćen raspored brodova.

8. u stanju *round*, igra šalje svim igračima poruku ROUND *x time* [*name1*; *name2*; ...; *namen*] gde *x* označava redni broj poteza (runde), *time* označava broj sekundi predviđenih za trajanje poteza, a *namex* označava ime x-tog igrača koji može da učestvuje u ovom potezu. Nakon slanja poruke, igra čeka vreme trajanja poteza tokom kojeg može od igrača da primi poruku:
   8.1. FIRE *id* [*T1C1*;*T2C2*;...;*TmCn*], gde *id* označava jedinstveni identifikator igrača, *Tx* označava tablu igrača ka kojoj je ispaljena granata, a *Cy* označava koordinatu u tabli. Primer izgleda poruke: FIRE 32814 [{arnold}0406;{bruce}1123;{sylvester}0000]
      8.1.1. u slučaju da je vreme za gađanje isteklo, da igrač čija se tabla gađa ne postoji, koordinate su van opsega ili ima više koordinata nego što igrač koji šalje poruku ima operativnih segmenata brodova, igra odgovara porukom FIRE_REJECTED
      8.1.2. u suprotnom, odgovara porukom FIRE_ACCEPTED
9. u stanju *update*, igra pronalazi uniju svih gađanja i saopštava porukom:
   9.1. UPDATE [T1C1S;T2C2S;...;TmCnS], gde *Tx* označava tablu igrača ka kojoj je ispaljena granata, *Cy* označava koordinatu u tabli, a S označava status koji može biti H (hit – pogodak) ili M (miss – promašaj). Primer izgleda poruke: UPDATE [{arnold}0406H;{bruce}1123M;{sylvester}0000H]
   9.2. Igračima čiji su svi brodovi potopljeni šalje poruku: GAME_OVER
   9.3. Ako preostane tačno jedan igrač sa najmanje jednim operativnim brodom:
      9.3.1. igraču koji je pobedio šalje poruku: VICTORY
      9.3.2. ostalim igračima šalje poruku: GAME_WON *name*, gde *name* predstavlja ime pobednika igre.
   9.4. Ako ne preostane nijedan igrač, svim igračima šalje poruku NO_VICTORY
   9.5. nakon slanja poruka iz tačke 9.1, i eventualno 9.2, igra čeka 3 sekunde i prelazi u stanje *round*, ako u igri ima više od jednog igrača sa operativnim brodovima.
   9.6. nakon slanja poruka iz tačke 9.3 ili 9.4 komunikacija sa igračima se završava.

10. u slučaju prijema komande koju ne prepoznaje, server vraća poruku ERROR *primljena_poruka*.

2. Funkcionalna specifikacija

U ovom odeljku data je funkcionalna specifikacija nekih od klasa koje treba da figuriraju u rešenju. Po potrebi treba napisati dodatne klase, koje nisu navedene ovde, a koje služe za međusobno povezivanje nekih od navedenih klasa. Paketi klasa, koje treba realizovati u okviru rešenja ovog projektnog zadatka, načelno treba da sadrže sledeće klase (odnosno klase koje realizuju odgovarajuću funkcionalnost):

common.Coordinate (koordinata)
Klasa koja predstavlja koordinatu u tabli za igru. Može da sastavi tekstualni opis koji odgovara formatu iz tačke 1 opisa komunikacionog protokola. Postoji uslužna metoda koja pravi nov objekat koordinata na osnovu teksta.

common.Ship (brod)
Klasa koja predstavlja brod. Stvara se sa zadatim brojem segmenata. Može da se orijentiše horizontalno ili vertikalno. Prvi segment broda (onaj sa najmanjim brojem vrste ili kolone) može

da se postavi na zadatu koordinatu. Može da sastavi tekst koji opisuje raspored segmenata po formatu iz tačke 6.1. Postoji uslužna metoda klase koja pravi nov brod na osnovu formata iz tačke 6.1. Može da utvrdi da li se preseca ili dodiruje (ivicom ili temenom) sa drugim brodom. Može da se onesposobi segment broda na zadatoj koordinati. Može da se sazna status segmenta broda na zadatoj koordinati.

common.Table (tabla)
Klasa koja predstavlja tablu za igru. Stvara se zadatih dimenzija. Sadrži brodove. Može da obriše svoj sadržaj. Može da rasporedi brod na zadatu koordinatu uz proveru uspešnosti. Može da dohvati brod čiji se segment nalazi na zadatoj koordinati. Može da utvrdi ukupan broj operativnih segmenata sadržanih brodova.

server.Player (igrač)
Aktivna klasa koja predstavlja igrača u igri (na serverskoj strani). Sadrži ime igrača i tablu za igru. Prilikom stvaranja se zadaje igra (videti u nastavku) u kojoj igrač učestvuje. Služi za komunikaciju sa klijentom predstavljenim klasom PlayerProxy (videti u nastavku). Svoju aktivnost ostvaruje tako što čeka da stigne poruka od klijenta kojom se označava željena aktivnost, u skladu sa prethodno opisanim protokolom. Po prispeću poruke pokuša njeno izvršenje i klijenta obavesti ishodu upućivanjem odgovarajuće poruke.

server.BattleOverseer (nadzornik bitke)
Aktivna klasa koja realizuje glavnu funkcionalnost igre koja predstavlja samu bitku. Pokrenuta bitka ulazi u stanje *deploy ships*, nakon kojeg ciklički ulazi u stanja *round* i *update*. U stanju *deploy ships* igra očekuje da igrači dojave raspored svojih brodova. Vreme za raspoređivanje je ograničeno, a počinje da se meri od trenutka kada su svi prijavljeni igrači potvrdili da su primili parametre igre. Smatra se da su igrači koji ne rasporede svoje brodove u predviđenom vremenu odustali od igre, ali ne bivaju automatski isključeni – mogu da učestvuju kao posmatrači. Igra je zadužena da vodi računa o konzistentnom stanju. Naime, zbog otvorenosti komunikacionog protokola, treba predvideti postojanje klijentskih aplikacija razvijenih sa malicioznim namerama. Takve klijentske aplikacije bi mogle da pokušaju obmanu u smislu nedozvoljenog raspoređivanja ili dimenzija brodova, nedozvoljenog broja gađanja u jednom potezu, itd.

Napomena: iako je broj navedenih stanja igre relativno mali, moguće je smisliti i druga, pomoćna (tranziciona) stanja. Zbog toga nije poželjno da rešenje ponašanja u datom stanju bude realizovano primenom selekcije (if/switch). Naprotiv, treba predvideti apstraktnu klasu koja predstavlja stanje igre i obezbediti apstraktnu metodu koja obezbeđuje ponašanje u datom stanju. Ponašanje igre u datom stanju, osim aktivnosti uključuje i skup prihvatljivih poruka koje mogu pristići od igrača. Deo ponašanja je zajednički: odbijanje nepoznate komande, odbijanje poruke neispravno identifikovanog igrača, itd. Konkretne klase stanja treba da realizuju onaj deo ponašanja koji je specifičan za dato stanje. Treba smisliti mehanizam kojim se bira naredno stanje, nakon što se tekuće završi. Razmotriti da li možda stanja treba da budu aktivne klase, a BattleOverseer da bude pasivna klasa?

server.Game (Igra)
Sadrži kolekciju igrača. Čitanjem sa standardnog ulaza može da se zada pristupna lozinka, može da se definiše veličina table za igru, broj i veličina brodova, vreme predviđeno za raspoređivanje brodova i vreme predviđeno za svaku rundu. Može da se zada najveći broj igrača, najmanje 2 igrača. Bitka može da se pokrene i da se prekine. Može da se traži prikaz table svakog igrača nakon svake runde. Može da pošalje poruku svim igračima uključenim u igru.

client.Player (Igrač)
Klasa koja predstavlja interfejs ka igraču. Na početku rada vrši povezivanje sa serverom na zadatoj adresi (adresa se zadaje bilo putem komandne linije pri pokretanju programa, bilo čitanjem sa standardnog ulaza nakon pokretanja programa). Zatim prikaže meni sa opcijama koje igraču stoje na raspolaganju: prikazivanje trenutnog stanja igre i napuštanje igre. Kada od servera stigne informacija o pripremi za početak bitke, prikaže se meni na kojem igrač može da kontroliše raspoređivanje svojih brodova i da javi da je raspoređivanje završeno. Tokom raspoređivanja, periodično se ispisuje preostalo vreme. Po prijemu obaveštenja od servera da počinje runda bitke, prikazuje se meni bitke, na kojoj igrač izdaje komande gađanja. Igraču treba omogućiti da vidi preostalo vreme tekuće runde, sadržaj tabli ostalih igrača, da sastavlja i menja sadržaj liste gađanja. Za komunikaciju sa serverom koristi se posebna klasa communication.Client (videti u nastavku). S obzirom na to da od servera informacije mogu asinhrono da stižu, komunikacija mora da se vrši u posebnoj niti.

3. Mrežna komunikacija

Na raspolaganju su **gotove klase** za mrežnu komunikaciju UDP protokolom, u paketu communication.

- CommunicationCommands: uslužna statička klasa (bez instanci) koja sadrži nabrojane komande predviđene komunikacionim protokolom. Potrebno je da studenti dopune ovu klasu do kompletnog skupa komandi predviđenih protokolom.
- SocketCommunicator: apstraktna klasa koja realizuje osnovne komunikacione operacije (slanje i primanje tekst poruke) u vezi server-klijent.
- Client: jedna vrsta SocketCommunicator-a koji se stvara sa zadatom adresom servera. Predviđeno je da se ova klasa koristi na klijentskoj strani za komunikaciju sa serverom.
- Server: jedna vrsta SocketCommunicator-a koji opslužuje proizvoljan broj klijenata (igrača) tako što preko definisanog komunikacionog porta prima poruke (naredbe) od klijenata i obrađuje ih. Komunikacija sa bilo kojim klijentom mora ići preko primerka ove klase. Server vodi evidenciju o svim prijavljenim klijentima. Svakom novom klijentu (koji uputi zahtev JOIN i zadovolji uslove servera) dodeljuje nov identifikacioni broj. Studenti treba da prošire ovu klasu tako da podrži sve naredbe predviđene komunikacionim protokolom.
- PlayerProxy: klasa koja ostvaruje komunikaciju između predstave igrača na serverskoj i pravog igrača na klijentskoj strani. Prilikom dohvatanja prispele poruke blokira pozivajuću nit. Zadatu poruku prosleđuje klijentu posredstvom serverskog komunikacionog objekta.

Napomene:
1. Neke klase iz paketa *server* i *client* su delimično implementirane, kao ilustracija, u cilju pojednostavljenja zadatka. Od studenata se očekuje da ove klase kompletiraju prema specifikaciji.
2. Od studenata se očekuje da one elemente specifikacije koji nisu dovoljno precizni sami dopune na razuman i opravdan način.
3. Izmena u specifikaciji komunikacionog protokola **nije dozvoljena**. Cilj je da serverske i klijentske komponete koje potiču od različitih studenata u potpunosti budu kompatibilne

u pogledu komunikacije. Dozvoljeno je proširivanje protokola, ali samo ako time nije narušena osnovna specifikacija.

4. Studenti mogu da menjaju (prvenstveno dopunjuju) implementaciju gotovih klasa iz paketa communication, vodeći računa o napomeni iz tačke 3.
5. Za izradu projektnog zadatka predviđen je timski rad, u timovima od po **dva** studenta. Jedan student treba da realizuje zajedničke klase (paket common) i klijentsku komponentu (program koji koristi igrač da bi igrao igru). Drugi student treba da realizuje serversku komponentu (program koji upravlja igrom).