Институт за математику и информатику
Природно-математички факултет
Универзитет у Крагујевцу

SINK SHIPS

семинарски рад

Професор:

др Ана Капларевић-Малишић

Студент:

Небојша Сретеновић 97/2015

Септембар 2017.

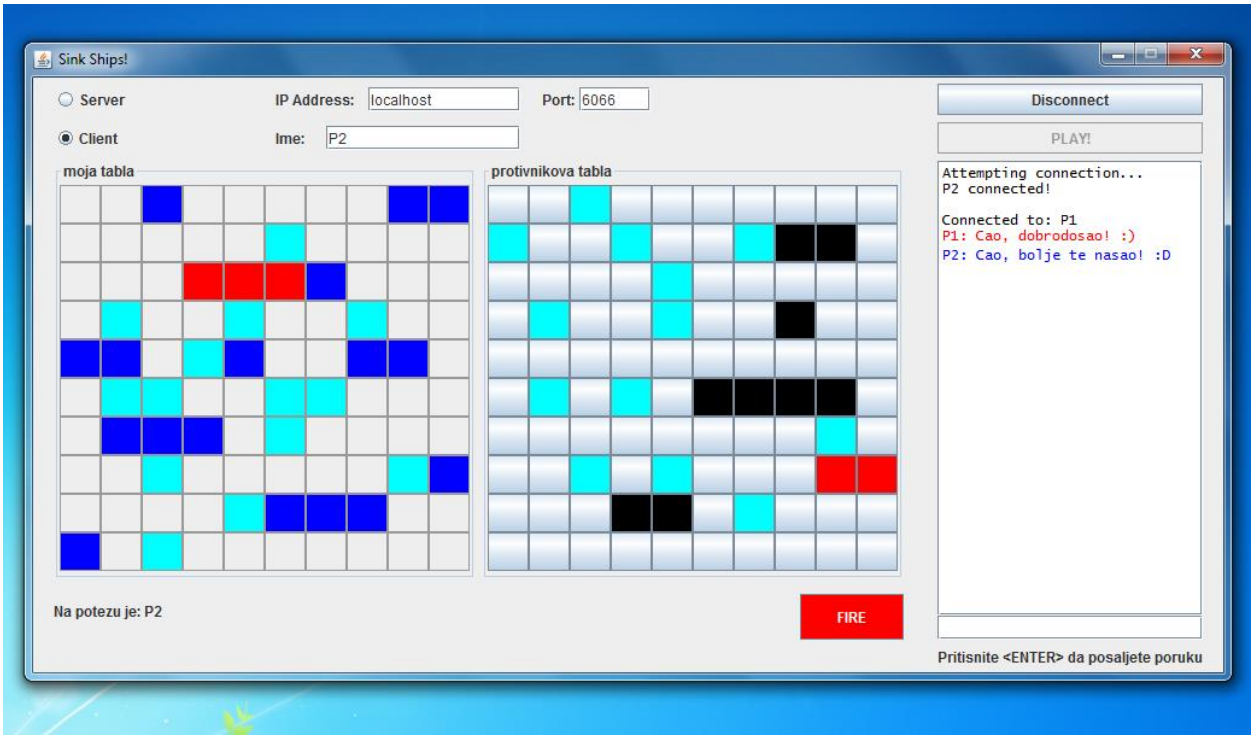
Садржај

1 Увод

Овај семинарски рад представља пратећи документ уз апликацију израђену као део испитних обавеза на предмету Објектно-оријентисано програмирање. Софтверско решење је развијено употребом Јава програмског језика и стандардних Јавиних библиотека и технологија. Резултат рада је игра *Sink Ships,* израђена по узору на чувену дечију игру *потапање бродића.*

1.1 О игри

Игра *Sink Ships* је табеларна потезна игра. Користе се 2 табле димензија 10x10 поља. Играчи најпре распореде по жељи 10 бродова, претежно различитих димензија, на својој левој табли. Након тога игра може да почне и играчи наизменично „гађају“ поља на десној табли која представља сакривену противникову таблу. Након „гађања“ поље се обоји одговарајућом бојом чиме играч бива обавештен да ли је погодио неки брод или не и ако јесте да ли је „потопио“ тај брод. Победник је онај ко први потопи све противникове бродове. У игри учествују 2 играча који се повезују са удаљених рачунара. Играчи могу да комуницирају путем чета са десне стране.



Слика 1 Приказ екрана игре

1.2 Спецификација захтева

Софтверско решење је реализовано тако да испуни следеће захтеве:

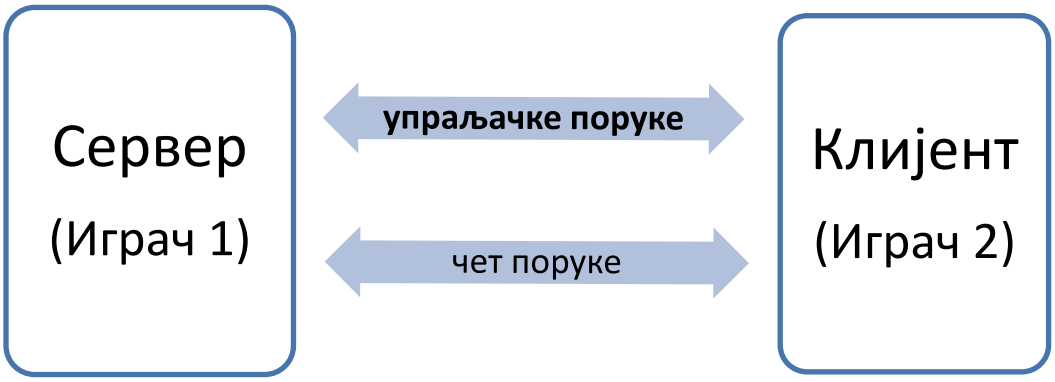
- Реализовати игру *Sink Ships* за два играча.
- Играчи играју један против другог, у наизменичним потезима.
- Омогућити играчима да комуницирају путем чета.
- Омогућити играчима да играју на одвојеним рачунарима, а да се комуникација одвија у мрежном окружењу.

2 Архитектура софтверског решења

Архитектура софтверског решења темељи се на клијент-сервер архитектури.

И **клијентска** и **северска** страна су готово равноправни ентитети у комуникацији и реализују следеће функционалности:

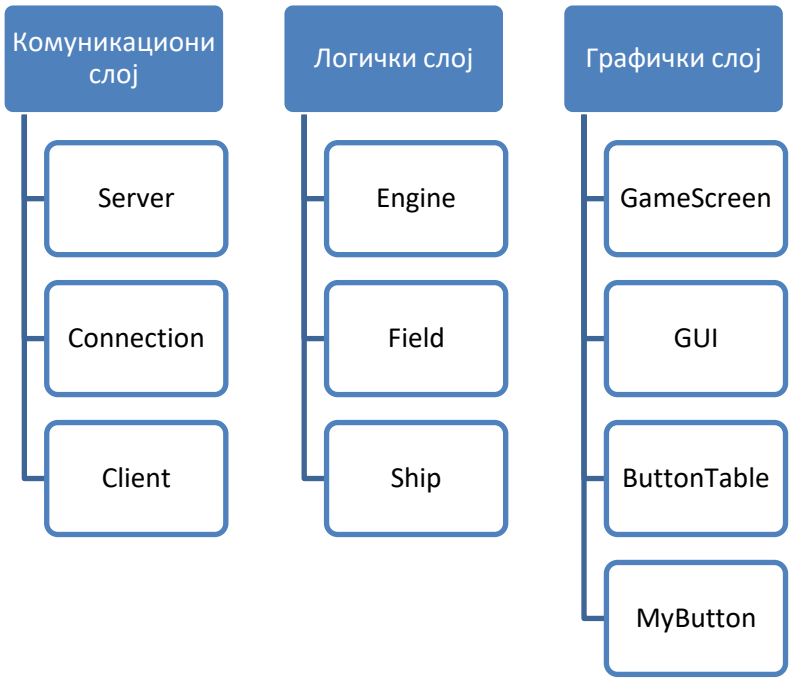
- Успостављање комуникационе везе ка другом ентитету (играчу)
- Слање и пријем порука којима се управља логиком игре
- Слање и пријем чет порука



Слика 2 Сервер-Клијент комуникација у игри

Софтверско решење садржи неколико функционалних целина реализованих у оквиру софтверских пакета које је могуће раздвојити у три слоја:

- **Графички слој** - задужен за приказ графичког интерфејса и интеракцију са играчима,
- **Логички слој** - задужен за реализацију логике игре,
- **Коминикациони слој** - задужен за остваривање комуникације између клијента и сервера.

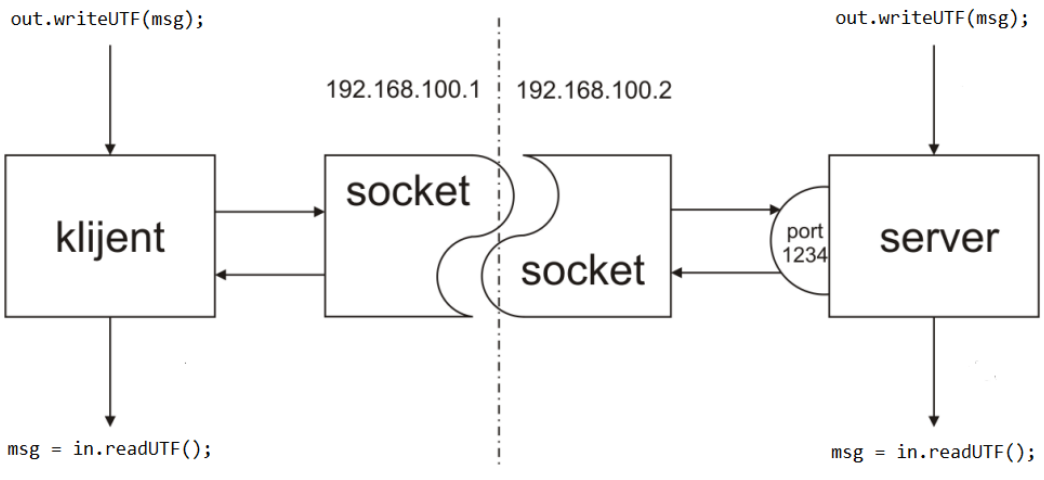


2.1 Комуникациони слој

Комуникација између клијента и сервера је остварена употребом сокета. Сокети представљају интерфејсе на мрежи преко којих се два уређаја „прикључују“, тиме успостављају комуникацију и начин размене података. Из угла програмера, сокет изгледа као нека врста „прикључка“. Када програмер жели да оствари комуникацију са удаљеним сервером, једноставно прикључи свој сокет са сокетом сервера и кроз тај прикључак се остварује размена података. Локални и удаљени сокети који су међусобно „прикључени“ носе име „сокет пар“ и карактеришу га четири податка:

- *IP* адреса и порт локалног сокета,
- *IP* адреса и порт удаљеног сокета.

Начин на који су клијент и север прикључени употребом сокета у игри *Sink Ships* илустрован је на слици 4.



Слика 3 Клијент-Сервер комуникација путем сокета

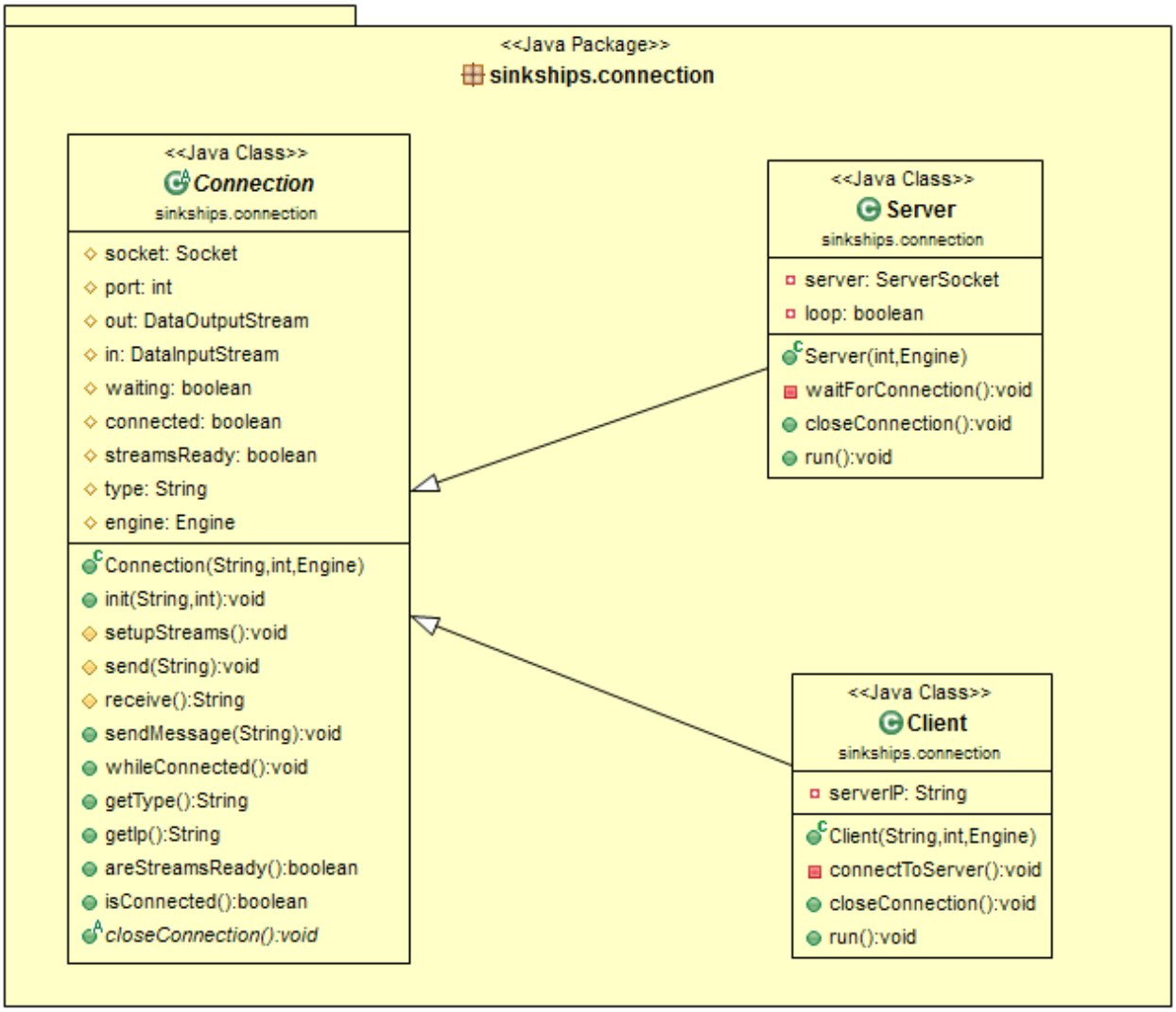
Јава имплементација сокета крије од програмера све оно што се налази испод сокета. Објекат сокета има своје улазне и излазне стримове. Стрим је објекат који може представљати извор или дестинацију података, и у зависности од тога разликујемо улазне (енгл. *input stream*) и излазне (енгл. *output stream*) стримове (на слици 4 *in* и *out*). Стримови у Јави су уређене секвенце бајтова неодређене дужине. Стримови подржавају различите типове података укључујући бајтове, примитивне типове података и објекте.

readUTF() – метод чита и смешта у *string* податак кодиран у *UTF-8* формату, а који је пристигао на *input stream*-у

writeUTF(msg) – метод шаље msg *string* податак кодиран у *UTF-8* формату на на *output stream*

2.1.1 Пакет *connection*

Класе које реализују комуникацију између клијента и сервера смештене су у пакету ***connection***.



Слика 4 Дијаграм класа пакета *connection*

Класа ***Connection*** је апстрактна, наслеђује класу *Thread* и обједињује заједничке податке и методе потребне и за сервер и за клијента. Ова класа јер генерализација *Server* и *Client* класа и садржи основне податке који се тичу конекције као што су број порта на ком се одвија комуникација, тип објекта конекције (*Server/Client*), а такође и методе за креирање стримова, слање и пријем порука са истих и уклањање стримова.

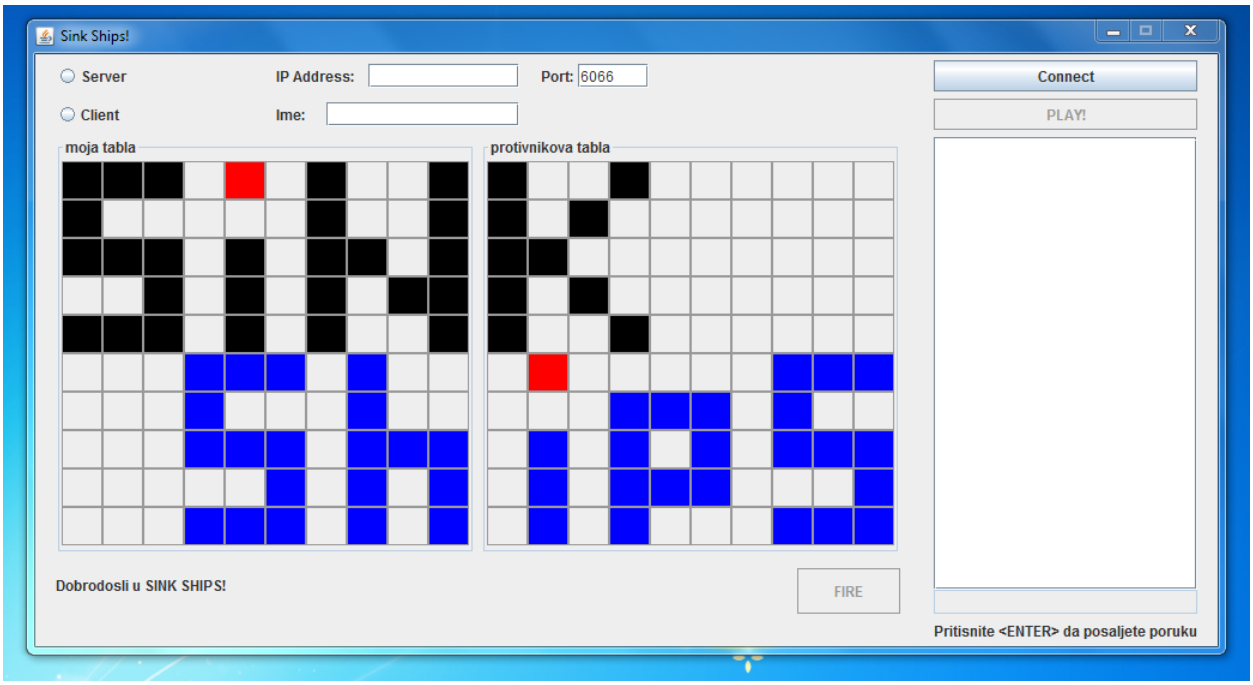
Класу *Connection* наслеђују класе *Server* и *Client.* Како су оне специјализације родитељске класе, у њима се налазе само њима својствени подаци и методе. ***Server*** креира серверски сокет на задатом порту. Након креирања ишчекује да се повеже клијент. ***Client*** за ту сврху садржи метод за повезивање на серверски сокет на задатој *IP* адреси и порту. Када се веза оствари, на обе стране покреће се петља која прима поруке и прослеђује логичком слоју на даљу обраду. Обе класе индиректно наслеђују класу *Thread* па се стога њихове финкционалности извршавају као засебне нити.

2.2 Графички слој

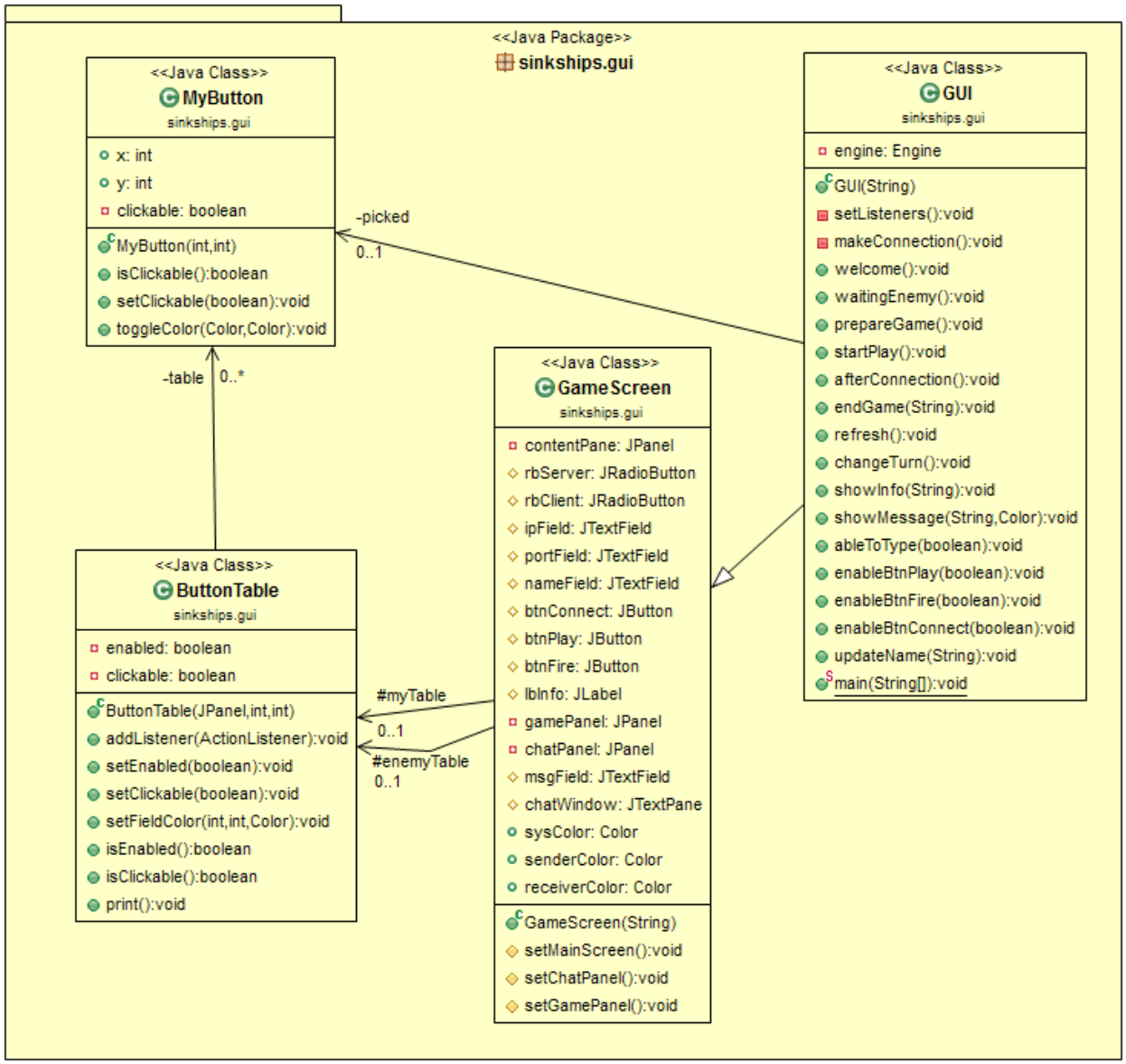
Графички слој апликације задужен је за приказ компоненти апликације које корисник види на екрану. За креирање графичких елемената користи се пакет ***gui*** и библиотека `javax.swing` .

Класа ***GameScreen*** служи да при покретању апликације исцрта и постави почетне параметре за све графичке компонетне као што су панел за подешавања, чет панел, и панел за игру са 2 табеле дугмића типа *ButtonTable* (садржи матрицу објеката класе *MyButton*).

Класа ***GUI*** наслеђује претходну класу и она је ту да визуелно спроводи динамику игре, односно да мења изглед графичких компоненти у складу са променама које диктирају корисник и логички слој. Она је такође централна класа апликације и у њој се налази *main* метод.



Слика 5 Поздравни екран апликације

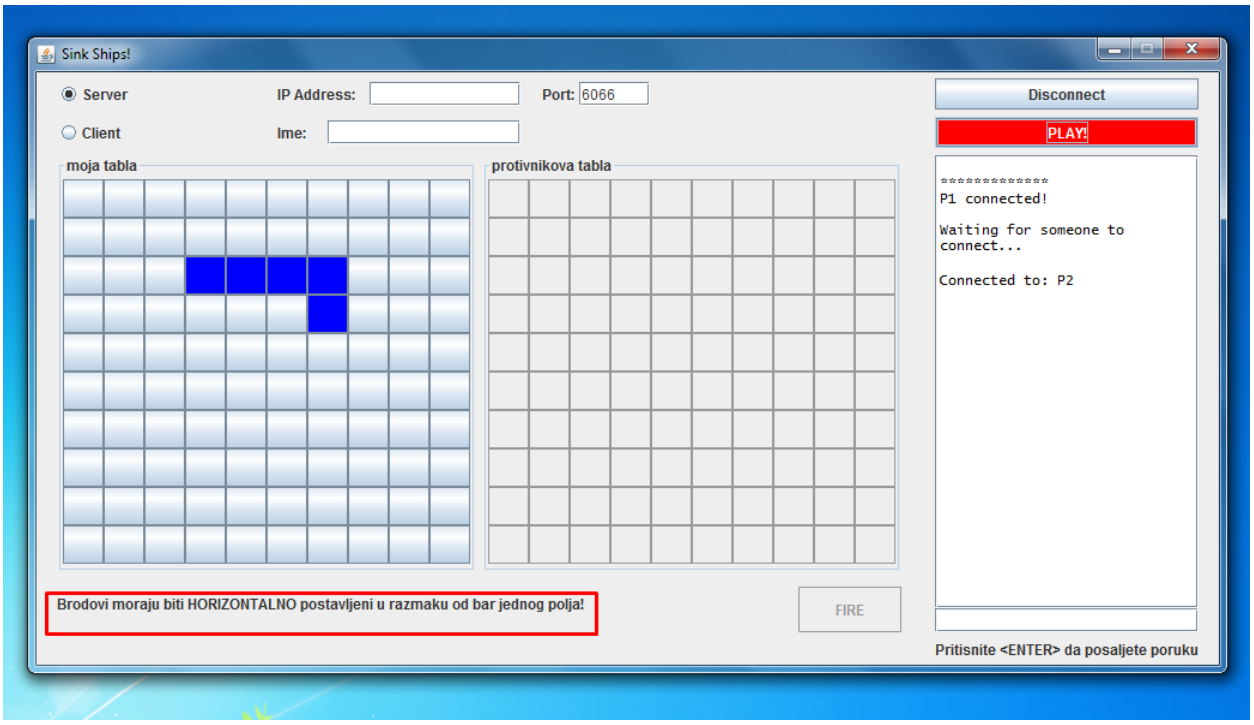


Слика 6 Дијаграм класа пакета *gui*

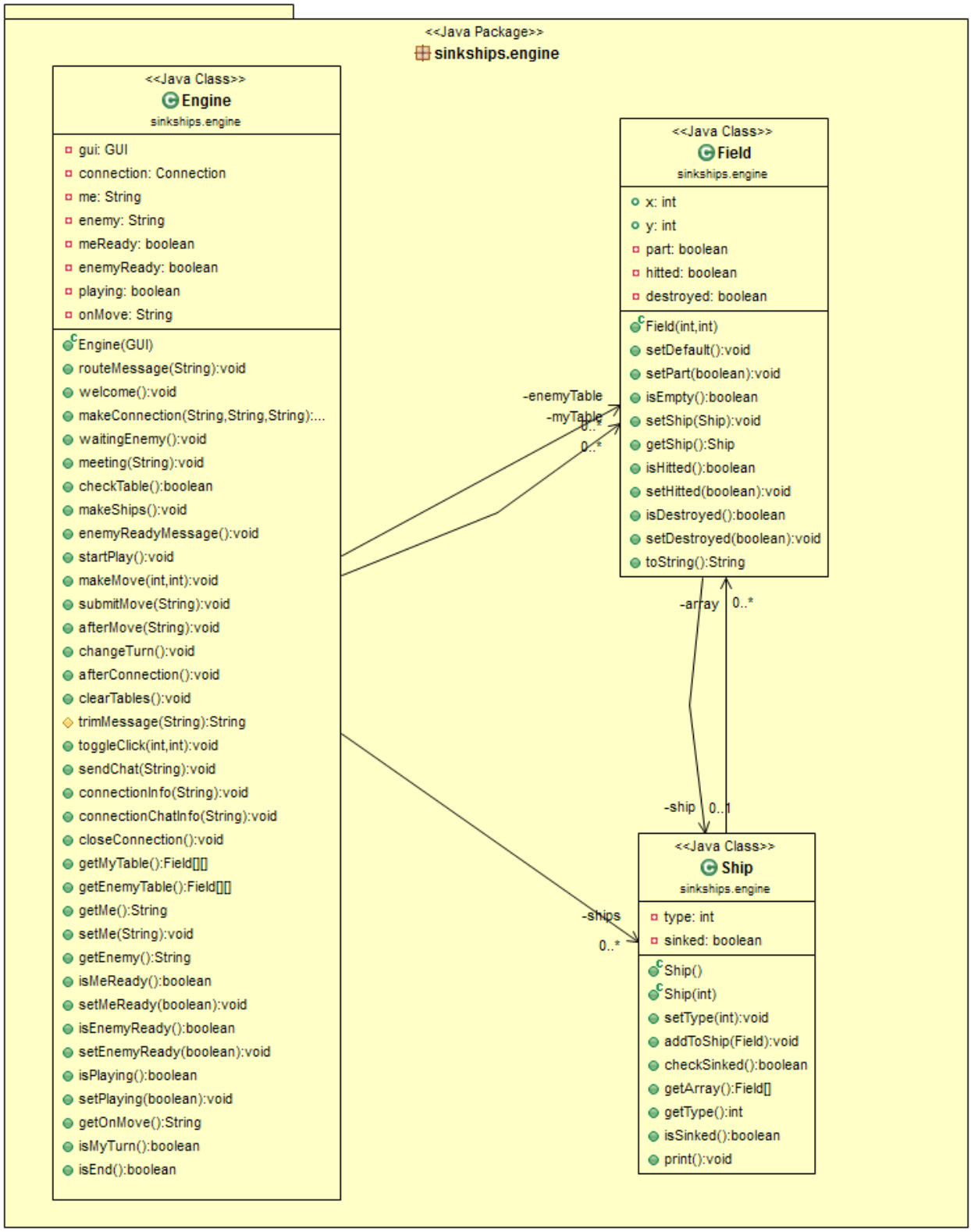
2.3 Логички слој

Језгро игре се реализује у логичком слоју кога сачињавају класе пакета ***engine.*** Класа ***Engine*** је главна класа овог слоја, постоји и код клијента и код сервера и служи да по успостављању конекције омогући:

- генерисање табли за игру и почетног поздравног екрана (слика 6),
- распоређивање бродова по жељи играча ,
- да се обавести играч уколико је неправилно поставио бродове (слика 8),
- генерисање задатих бродова и смештање истих у низ,
- слање играчевог потеза и пријем резултата истог у виду кодирано управљачке поруке,
- пријем противниковог потеза и слање резултата у виду кодиране управљачке поруке,
- ажурирање стања обе табле и бродова у низу,
- редовну проверу да ли је дошло до уништења свих бродова (краја игре).



Слика 7 Обавештење о неправилном постављању бродова



Слика 8 Дијаграм класа пакета *engine*

2.3.1 Класа *Field*

Класа ***Field*** се корсти као поље за прављење „моје“ и „противникове“ табле димензија 10x10 поља. Свако поље има *x* и *y* координате и атрибуте који описују стање поља којим се такође описујује и стање бродова.

Могућа стања су:

- Део брода (*True/False*),
- Погођено (*True/False*),
- Уништено (*True/False*).

Стања се комбинују како би се прецизно описало стање сваког брода и изабрала одговарајућа боја за дугмиће у *GUI*-ју . Комбинације могу бити:

- Није део брода, није погођено (без боје)
- Није део брода, погођено (светло плава)
- Део брода, није погођено (тамно плава)
- Део брода, погођено, није уништено (црвена)
- Део брода, погођено, уништено (црна)

2.3.2 Класа *Ship*

Класа ***Ship*** представља брод на табли, односно низ суседних поља(*Field*) која чине брод. Брод се може састојати од минимум једног и максимум 4 поља. Поред броја типа (1-4) брод садржи информацију о томе да ли је потопљен или не и метод којим се то проверава.

2.3.3 Категорије порука

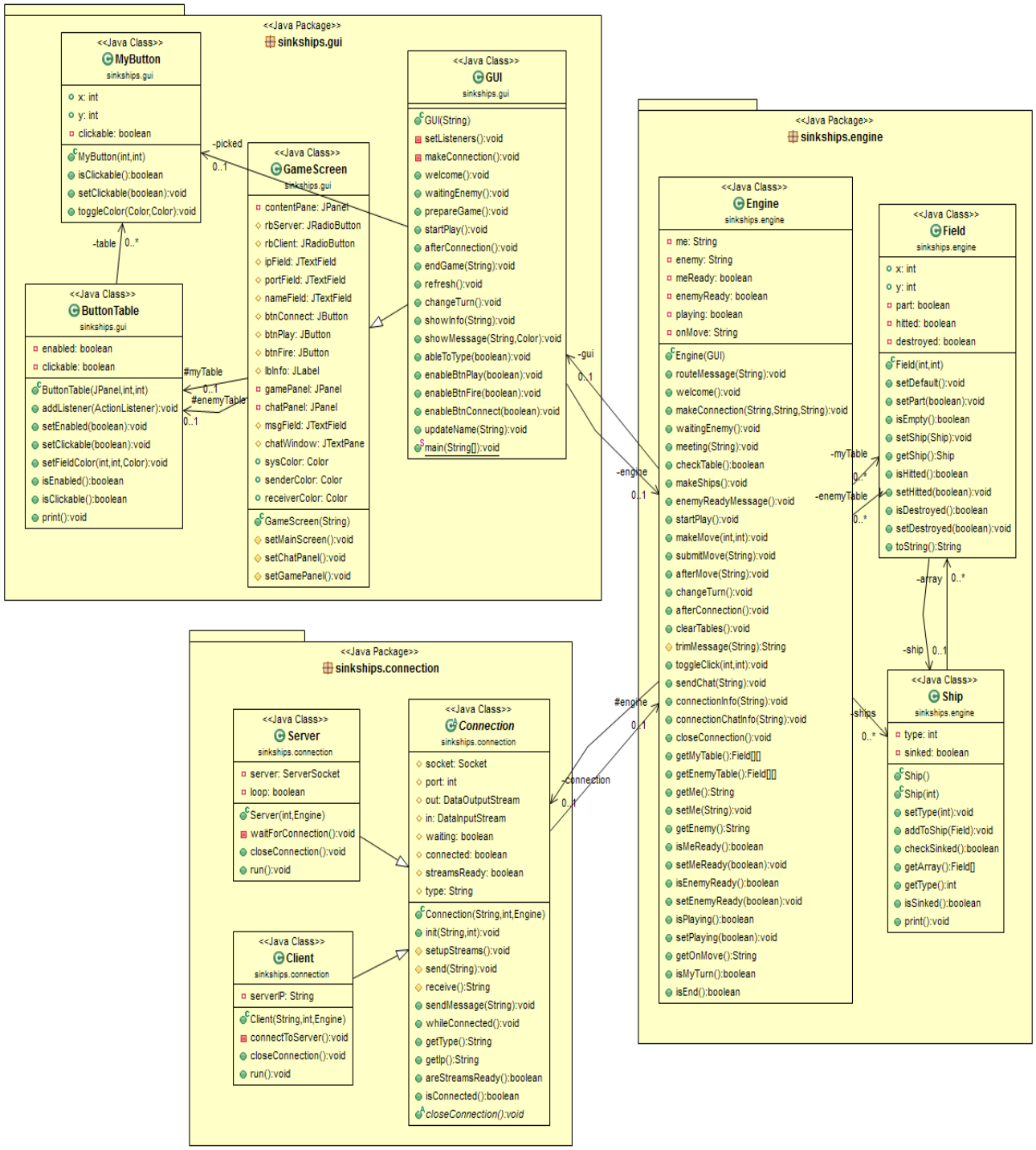
Поруке које представљају податке типа *String,* главни су носиоци радње у игри. Свакој поруци се пре слања додаје *String* заглавље које дефинише њен тип. Касније када порука стигне и проследи се *engine*-у, он на основу типа може да зна коју функцију следећу да примени. Поруке могу добити следећа заглавља:

- **#chat#** (чет порука)
- **#meeting#** (слање имена другом играчу – упознавање)
- **#ready#** (обавештење да је играч поставио бродове и да је спреман за игру)
- **#move#** (потез, кординате гађаног поља)
- **#dmginfo#** (одговор на потез, информација о штети)
- **#end#** (обавештење да су сви бродови уништени и да је крај игре)

```
33
34      public void routeMessage(String msg){
35          System.out.println("engine.routeMessage()");
36
37          if(msg.startsWith("#chat#"))
38              gui.showMessage("\n"+enemy+": "+trimMessage(msg), gui.receiverColor);
39
40          else if(msg.startsWith("#move#"))
41              submitMove(trimMessage(msg));
42
43          else if(msg.startsWith("#dmginfo#"))
44              afterMove(trimMessage(msg));
45
46          else if(msg.startsWith("#meeting#"))
47              meeting(trimMessage(msg));
48
49          else if(msg.startsWith("#ready#"))
50              enemyReadyMessage();
51
52          else if(msg.startsWith("#end#"))
53              gui.endGame(me);
54      }
55
```

Слика 9 routeMessage метода за усмеравање порука

2.4 Дијаграм класа свих пакета



Слика 10 Дијаграм класа свих пакета

3 Финалне напомене

3.1 Минимални системски захтеви

Минимални системски захтеви:

- Оперативни систем: *Windows*, *Linux*, (*Mac* - није тестирано).
- jpe: 1.8.0 (*Java Runtime Environment*)

Да би игра започела, неопходно је да се један играч повеже као сервер, а други као клијент. Сервер чека конекцију и након што се пријави један играч (клијент), остале пријаве на сервер биће онемогућене. Сваки наредни играч који покуша да се повеже на сервер неће успети.

3.2 Планирана унапређења

Планирана унапређења су:

- Могућност постављања бродова вертикално
- Боље навођење корисника при постављању бродова
- Бољи дизајн панела за информације
- Звукови
- Исправка багова

Литература

[1] https://imi.pmf.kg.ac.rs/moodle/course/view.php?id=33,

[2] https://www.youtube.com/watch?v=KdTsY3G_To0,

[3] https://imi.pmf.kg.ac.rs/moodle/pluginfile.php/5858/mod_resource/content/0/Predavanja/RM-Predavanje-9.pdf,

[4] https://www.youtube.com/watch?v=pr02nyPqLBU,

[5] https://www.tutorialspoint.com/java/java_networking.htm,

[6] http://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html,

[7] http://docs.oracle.com/javase/tutorial/uiswing/events/windowlistener.html,

[8] https://imi.pmf.kg.ac.rs/moodle/pluginfile.php/8466/mod_resource/content/1/seminarskiOOP%20Jovica.pdf,