Object Georiënteerd Ontwerp

Zeeslag

Yanice Slegers

Kevin Peelman

Louis Roeben

Dennis Stallaert

2de fase Toegepaste Informatica

21 december 2016

University Colleges Leuven-Limburg
Campus Proximus
Geldenaaksebaan 335
3001 Heverlee
Tel. +32 (0) 16 375 700

Inhoud

Algemene opmerking

Dit verslag gaat over de opdracht ‘Zeeslag’, dat zal meetellen voor 5 punten van je totaalscore van dit vak. Dit verslag vormt de toelichting bij jouw code.

Je wordt geacht om je voor het verslag volledig aan deze template te houden. Buiten deze ‘algemene opmerking’ zijn alle hoofdingen verplichte onderdelen van het verslag. Je print dit verslag af (dubbelzijdig) -1 exemplaar per groep en levert dit in bij de start van labo OOO op woensdag 21/12

EXTRA: Je bent verplicht om je in code gebruik te maken van het @author commentaarveld!

Je maakt een zip file van als je source code (.java bestanden, geen .class bestanden) van alle klassen die van belang zijn voor 000 opdracht. Je voegt tevens de laatse versie van je verslag(word document) toe aan deze zip file. Je mailt deze zip file naar patrick.fox@ucll.be uiterlijk voor zondag 25/12 23.59 uur. Naam van de zip file:Zeeslag_UCLL gevolgd door familienamen van groepsleden gescheiden door underscore (vb Zeeslag_UCLL_Dox_Lenaerts_Jansen.zip)

Tevens maak je van je zzeslag oplossing een jar file die kan runnen onder Java8. Naam van de jar file is dezelfde als naam zip file maar met extentie .jar vb Zeeslag_UCLL_Dox_Lenaerts_Jansen.jar). Deze jar file mail je samen met je zip file naar patrick.fox@ucll.be

Het spreekt voor zichzelf dat de doorgemailde code overeenstemt met de gecommitted code in je repository op projectwerk.khleuven.be

Vereisten

Geef voor het gehele project aan welke vereisten (evt. zelf verder uitgewerkt/opgesplitst aan de hand van de opgave) je succesvol hebben geïmplementeerd, en welke topics niet gelukt zijn. Indien je een deel van de voorziene vereisten niet afgewerkt hebt, geef dan aan waarom niet… De reden kan “tijdsgebrek” zijn, het kan een issue zijn “wist niet hoe, het crashte”, of het kan zijn dat je een zeer goede reden had om het niet te implementeren…

Voeg het definitieve gegenereerde klassendiagramma van je code toe, als afzonderlijke afbeelding, als bijlage bij dit rapport.

Als er zaken zijn uit de opdracht die je niet hebt kunnen uitwerken of die je beter zou willen uitwerken, dan som je deze hier tevens op (in laatste rij van tabel)

User story	OK?	Issues? (indien niet ok)
1. Teken bord	Ja	
2. Plaats schepen	Ja	
3. Controle plaats schepen	Ja	
4. AI speler plaatst schepen		
5. Start spel	Nee	We hebben geen State patroon gebruikt voor het verschil tussen een nieuw en gestart spel te maken. We gebruiken state bij targets. Meer uitleg kan u vinden bij het overlopen van de verschillende patterns.
6. Aanvallen		
7. Schip geraakt		
8. Aangevallen worden		
9. Toon score		
10. Einde spel		
Extra stories?		

Patterns

Geef voor elk gezien patroon aan waar je het gebruikt hebt (mogelijkerwijs meer dan eens). Genereer een klassendiagram voor elke situatie waarin je het patroon gebruikt hebt.

Geef extra informatie (voordelen / waarom / …). Als je een patroon niet toegepast hebt, leg je uit waarom niet.

	Toegepast (ja/nee) In welke stories(nr) Waarom toegepast(voordeel)	Bijhorend klassendiagram (uit object aid)
Observer	We gebruiken dit in user story 9. Wanneer een speler aanvalt roept dit een update functie aan. Deze functie geeft de score door aan de view die deze toont achter de speler zijn naam. Dit heeft als voordele dat de view automatisch wordt aangepast wanneer er een bepaalde actie heeft plaatsgevonden. (De speler klikt, de AI stuurt een target.)	Nog niet geïmplementeerd.
Strategy	We gebruiken Strategy voor onze AI. Deze heeft de keuze om een simpel en advanced vorm aan te nemen. Dit was handig omdat de plaats en aanval strategie gelijkaardige acties hebben maar een andere, soms moeilijkere implementatie moeten hebben. Zo dupliceren we geen code. Het is nu ook eenvoudiger om nieuwe strategies toe te voegen aan het spel. Deze moeten dan met relatief minder moeite worden geïmplementeerd omdat we programmeren	UML 9

	naar een interface.	
Simple Factory	We gebruiken simple factory onder ander voor schepen en targets (een deel van een schip) aan te maken.	UML 6
Façade	We gebruiken facades voor vrijwel alle packages waaronder de Ai en ship repository.	UML 5
Singleton	Ja, story 1,2,3 er mag maar één soort van deze klassen worden aangemaakt. We gebruiken deze klassen om bepaalde settings van het spel op te slaan. (Groote, aantal schepen van een bepaald type,…)	UML 2
State	We gebruiken de State strategie in verschillende story's. Onder andere in user story 1, 2 en 7. Wanneer we een veld aanklikken verandert dit target van state. Wanneer een schip geplaats wordt dit field healty, als een deel van een schip geraakt wordt krijgt dit een damaged state. Als het schip zinkt een sunken state,… Een van de voordelen hiervan is dat de state at runtime kan worden veranderd en dat we op voorhand kunnen definieren naar welke state men moet veranderen wanneer men geraakt wordt.	UML 8

MVC	We hebben dit principe over de stories heen gebruikt. Onze view kent enkel maar de controller en onze klasse’s in de model kunnen enkel maar interactie (waar nodig) hebben met de controller. Het voordeel hiervan is dat de controller werkt als een soort van facade voor de view. We kunnen hier complexiteit verbergen voor de user die enkel maar acties doet zoals, klikken, een item selecteren,…	kleurenoverzicht
Andere…		

Speciale topics

Geef voor elke “speciale topic” aan of je het gebruikt hebt of niet, en zo ja, waar. Toon eventueel aan aan met een klassendiagramma.

	Toegepast (ja/nee) In welke stories(nr) Waarom toegepast(voor deel)	Bijhorend klassendiagram (uit object aid) of extra uitleg
Enum	We gebruiken dit onder andere in story 2,3,4. We hebben via enum de verschillende schepen geïmplementeerd. Zo konden we makkelijk alle type schepen oproepen, vergelijken, …	UML 6
Properties	De Ai package gebruikt een properties bestand voor het bepalen van behavior.	
Reflection	Reflection wordt gebruikt in de Ai package en verscheidene factories	
Mockito testing		
Andere…		

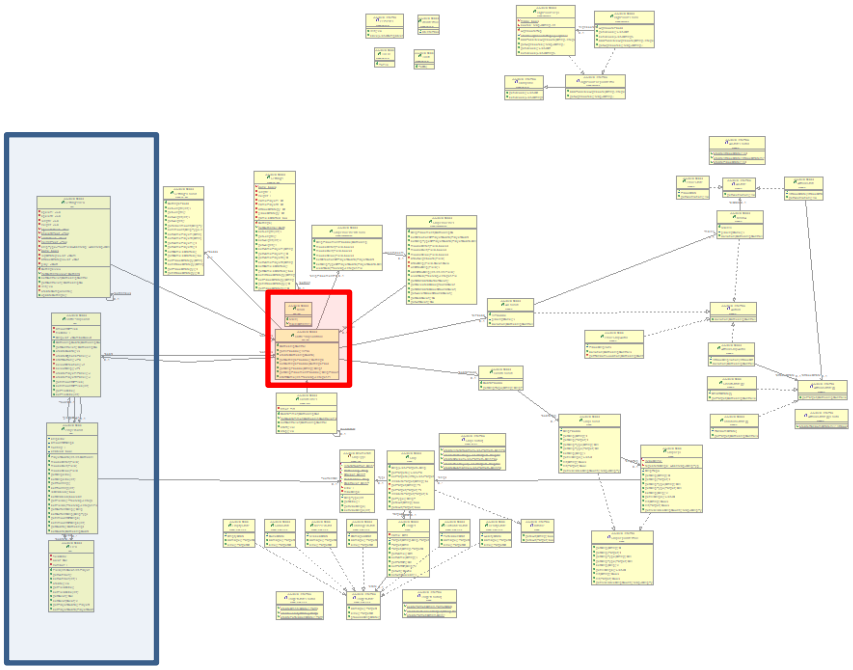
Werkverdeling

Geef aan in percentages hoeveel je bij benadering gespendeerd hebt aan deze opdracht

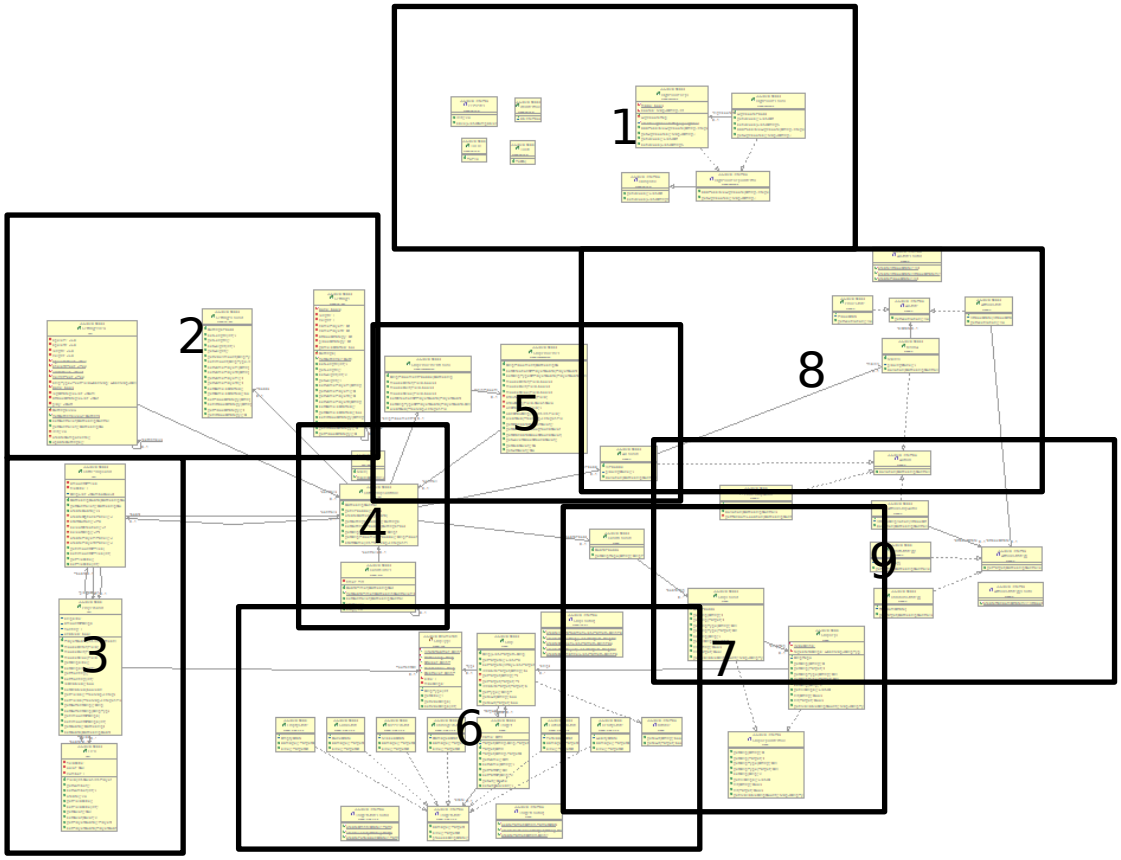
	Yanice Slegers	Kevin Peelman	Louis Roeben	Dennis Stallaert
Ontwerp	30	30	40	0
Klassendiagrammen	33	33	33	0
Implementatie	33	33	33	0
Verslag	50	50	0	0

Klassediagramma

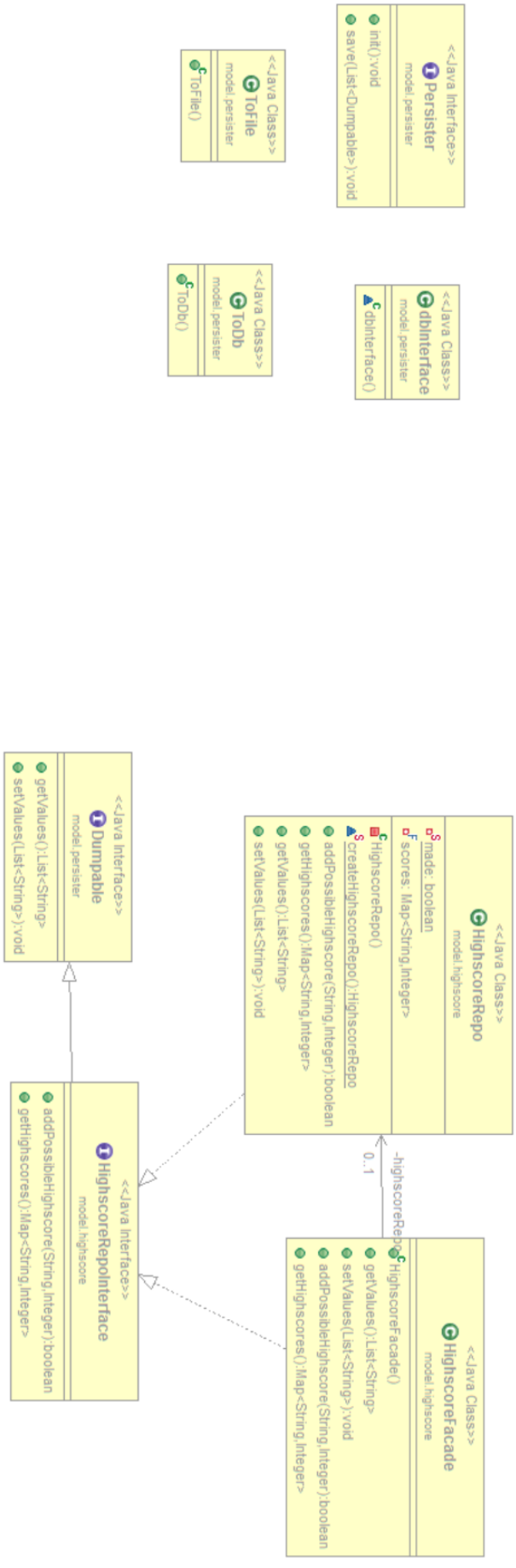
UML, view is blauw, controller is rood, model is de rest.



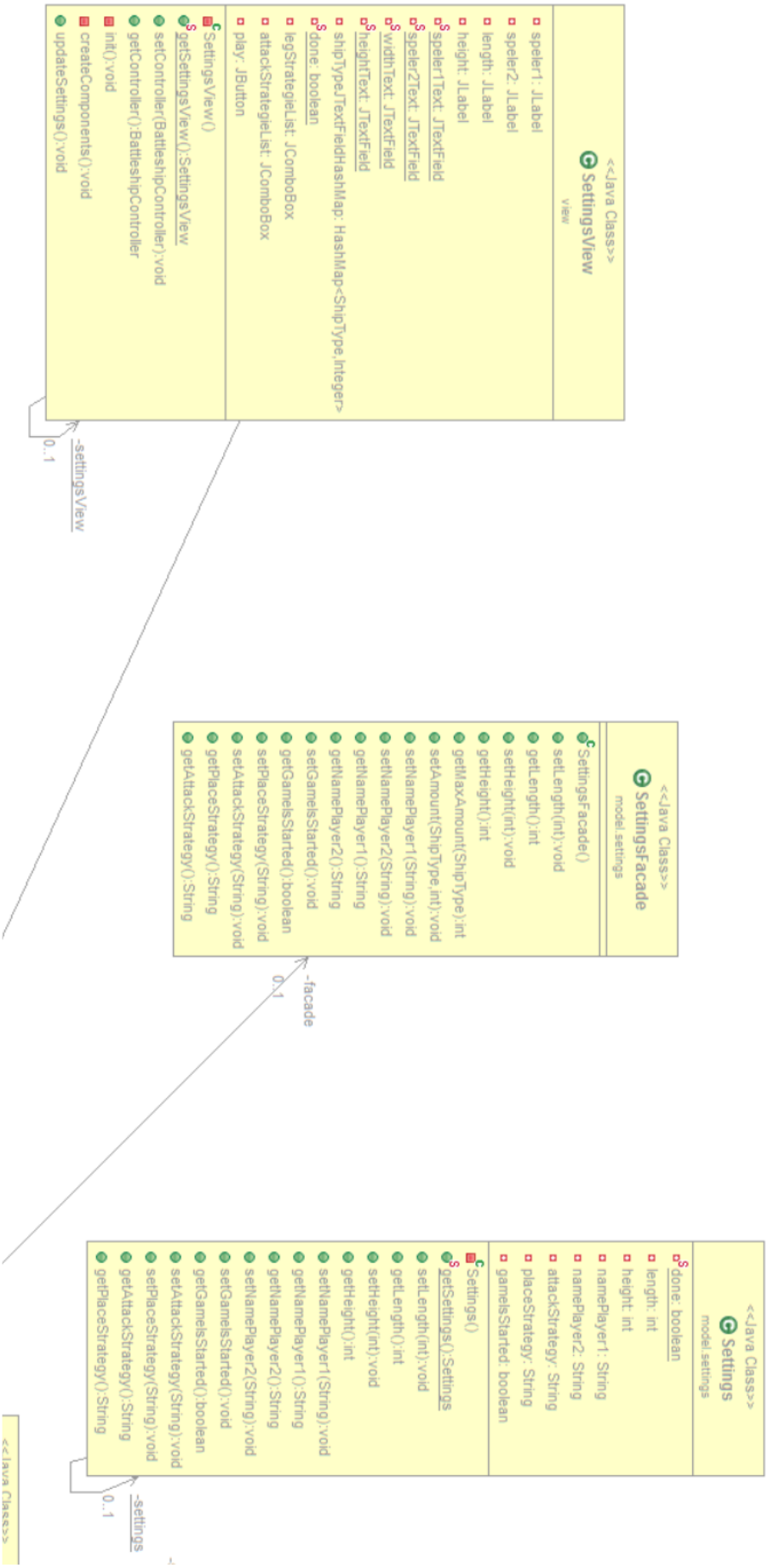
Overzicht van de verdere opdeling klassendiagrammas.



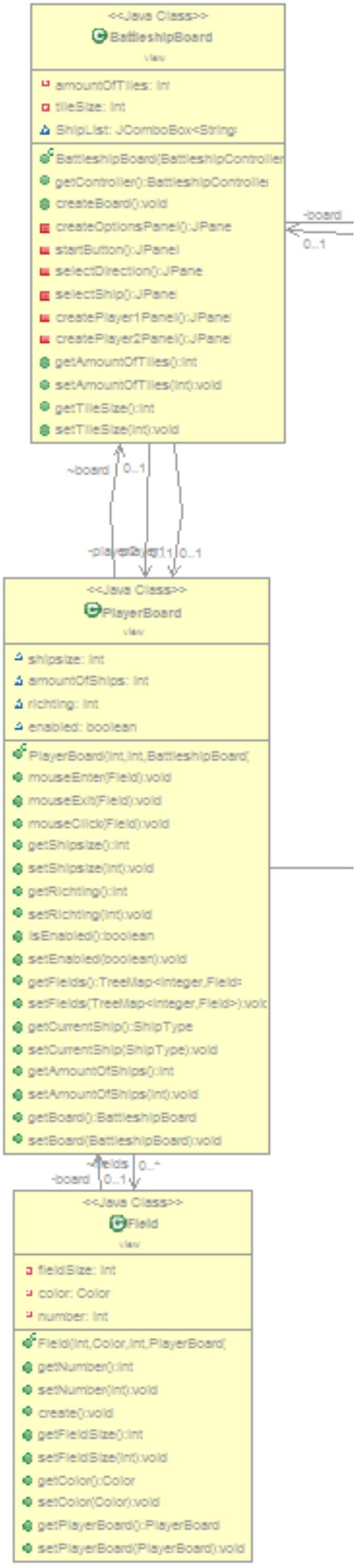
1 Database + highscores



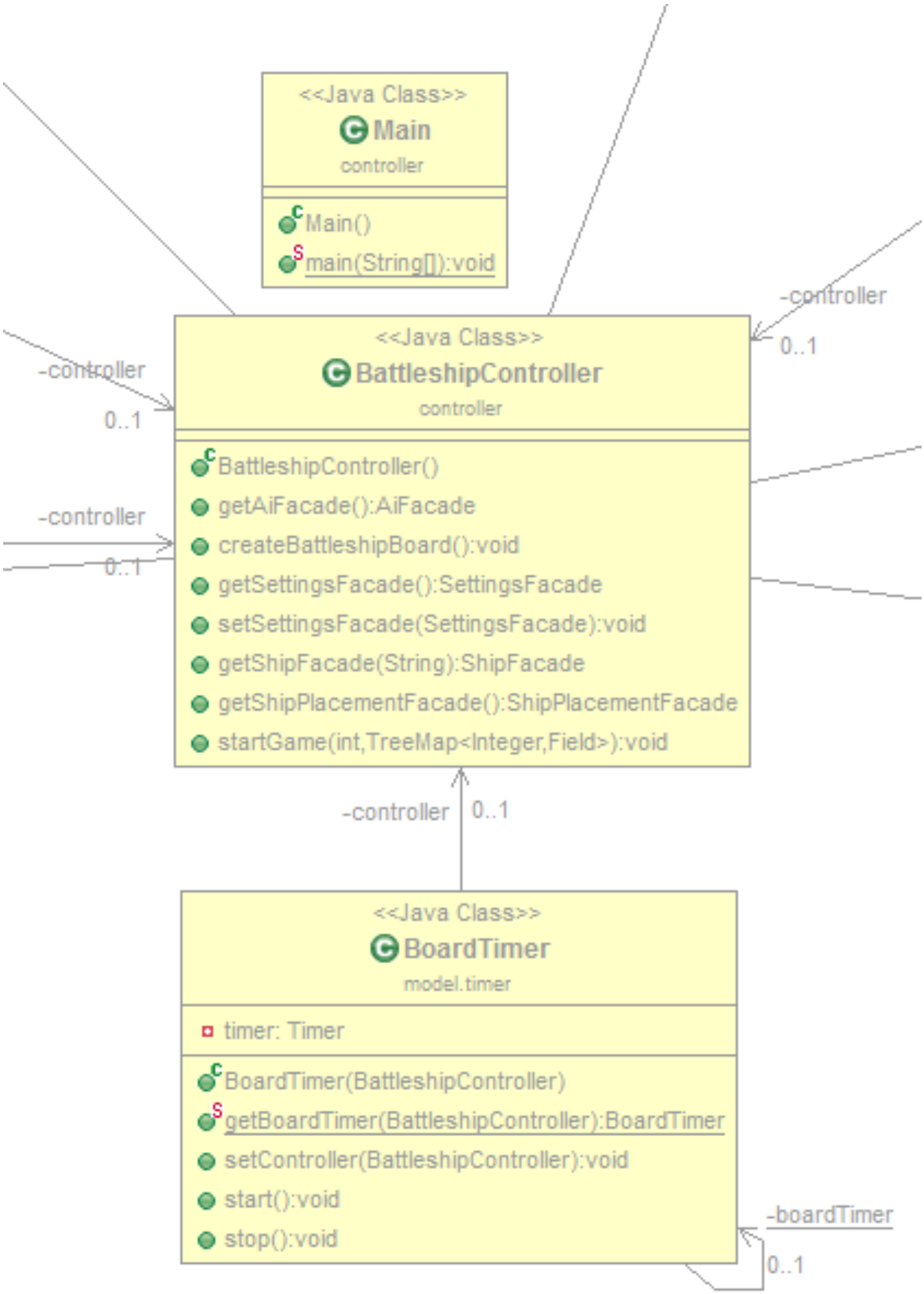
2 Settings


«Java Class»
SettingsView
view
speler1: JLabel
speler2: JLabel
length: JLabel
height: JLabel
speler1Text: JTextField
speler2Text: JTextField
widthText: JTextField
heightText: JTextField
shipTypeJTextFieldHashMap: HashMap<ShipType,Integer>
done: boolean
legStrategieList: JComboBox
attackStrategieList: JComboBox
play: JButton
SettingsView()
getSettingsView():SettingsView
setController(BattleshipController):void
getController():BattleshipController
init():void
createComponents():void
updateSettings():void
-settingsView
0..1
«Java Class»
SettingsFacade
model.settings
SettingsFacade()
setLength(int):void
getLength():int
setHeight(int):void
getHeight():int
getMaxAmount(ShipType):int
setAmount(ShipType,int):void
setNamePlayer1(String):void
setNamePlayer2(String):void
getNamePlayer1():String
getNamePlayer2():String
setGameIsStarted():void
getGameIsStarted():boolean
setPlaceStrategy(String):void
getPlaceStrategy():String
setAttackStrategy(String):void
getAttackStrategy():String
-facade
0..1
«Java Class»
Settings
model.settings
done: boolean
length: int
height: int
namePlayer1: String
namePlayer2: String
attackStrategy: String
placeStrategy: String
gameIsStarted: boolean
Settings()
getSettings():Settings
setLength(int):void
getLength():int
setHeight(int):void
getHeight():int
setNamePlayer1(String):void
getNamePlayer1():String
getNamePlayer2():String
setNamePlayer2(String):void
setGameIsStarted():void
getGameIsStarted():boolean
setAttackStrategy(String):void
setPlaceStrategy(String):void
getAttackStrategy():String
getPlaceStrategy():String
-Settings
0..1
«Java Class»

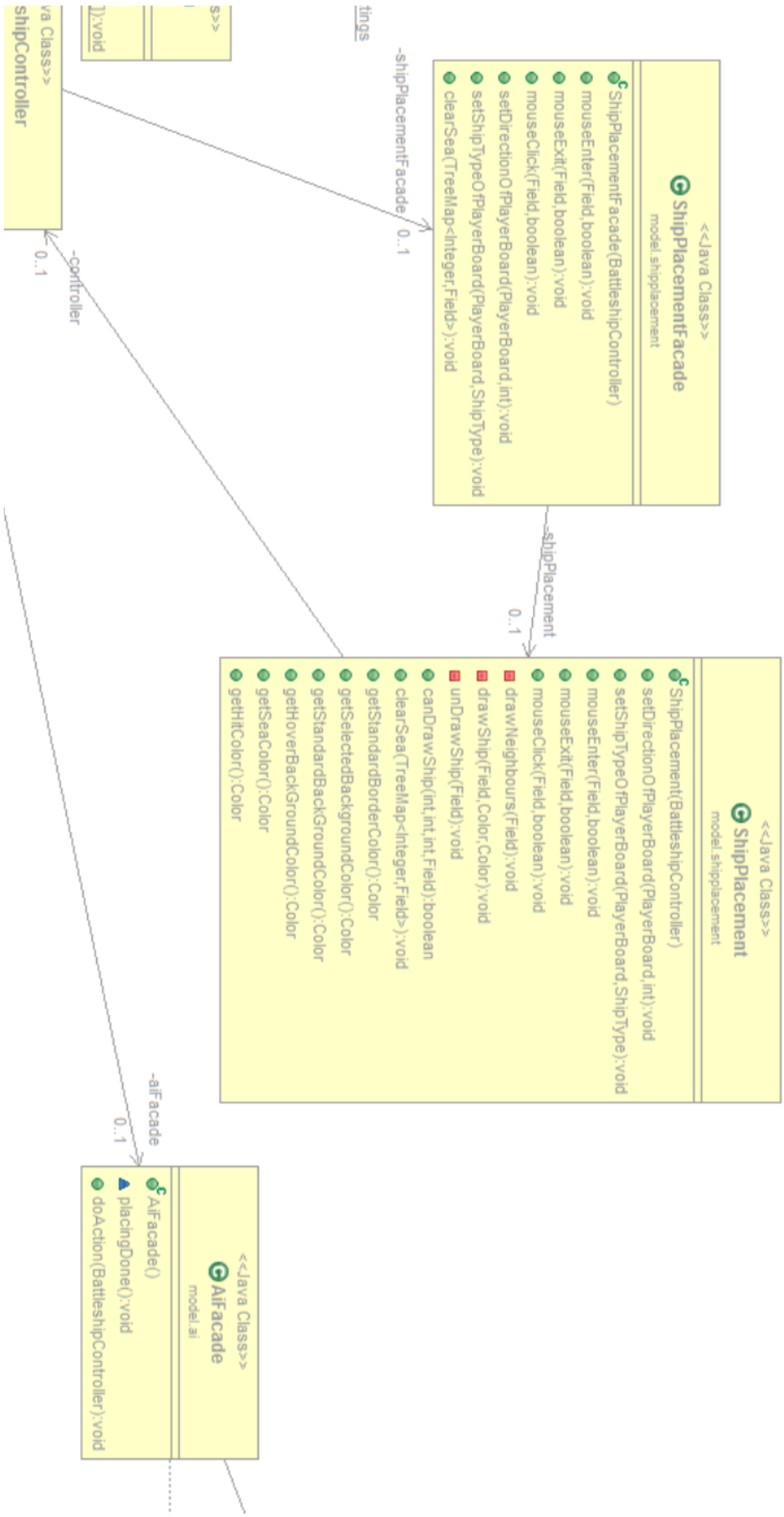
3 View


<<Java Class>>
BattleshipBoard
view
amountOfTiles: Int
tileSize: Int
ShipList: JComboBox<String>
BattleshipBoard(BattleshipController
getController():BattleshipControlle
createBoard():void
createOptionsPanel():JPane
startButton():JPanel
selectDirection():JPane
selectShip():JPanel
createPlayer1Panel():JPane
createPlayer2Panel():JPane
getAmountOfTiles():Int
setAmountOfTiles(Int):void
getTileSize():Int
setTileSize(Int):void
-board
0..1
~board
0..1
0..1
<<Java Class>>
PlayerBoard
view
shipsize: Int
amountOfShips: Int
richting: Int
enabled: boolean
PlayerBoard(Int,Int,BattleshipBoard(
mouseEnter(Field):void
mouseExit(Field):void
mouseClick(Field):void
getShipsize():Int
setShipsize(Int):void
getRichting():Int
setRichting(Int):void
isEnabled():boolean
setEnabled(boolean):void
getFields():TreeMap<Integer,Field>
setFields(TreeMap<Integer,Field>):voi
getCurrentShip():ShipType
setCurrentShip(ShipType):void
getAmountOfShips():Int
setAmountOfShips(Int):void
getBoard():BattleshipBoard
setBoard(BattleshipBoard):void
~fields
0..*
-board
0..1
<<Java Class>>
Field
view
fieldSize: Int
color: Color
number: Int
Field(Int,Color,Int,PlayerBoard(
getNumber():Int
setNumber(Int):void
create():void
getFieldSize():Int
setFieldSize(Int):void
getColor():Color
setColor(Color):void
getPlayerBoard():PlayerBoard
setPlayerBoard(PlayerBoard):void

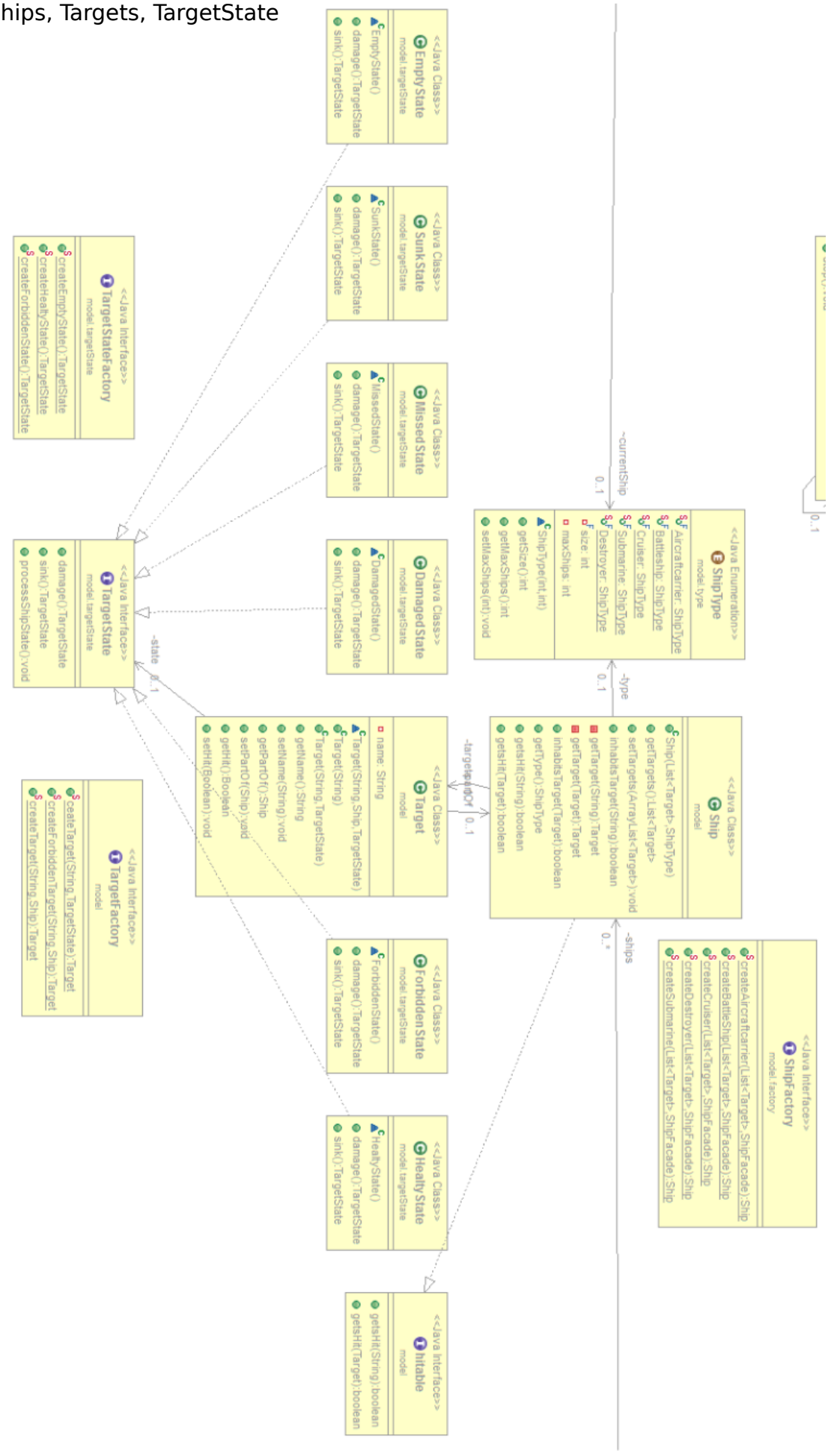
4 Controller



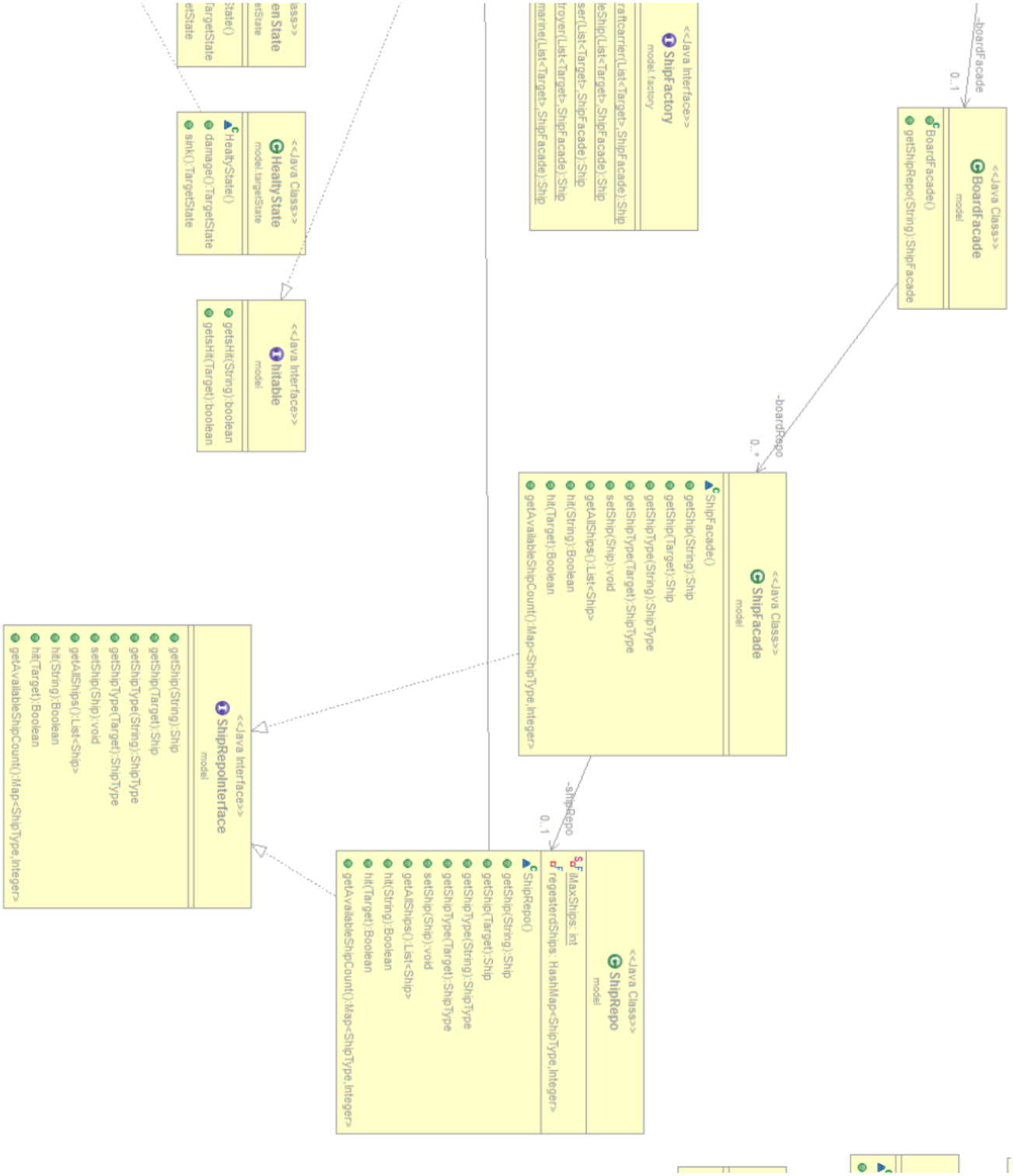
5 ShipPlacement



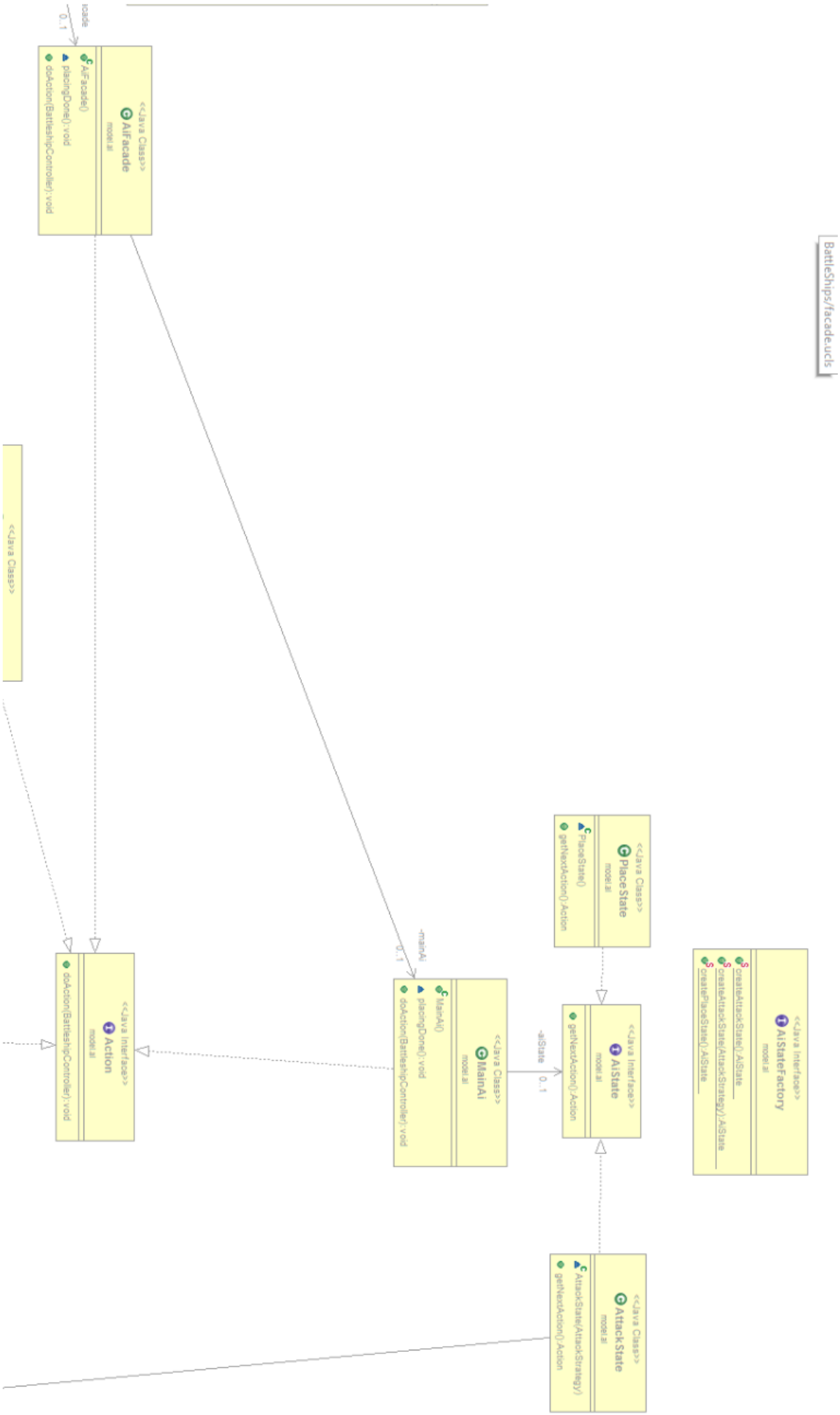
6 Ships, Targets, TargetState



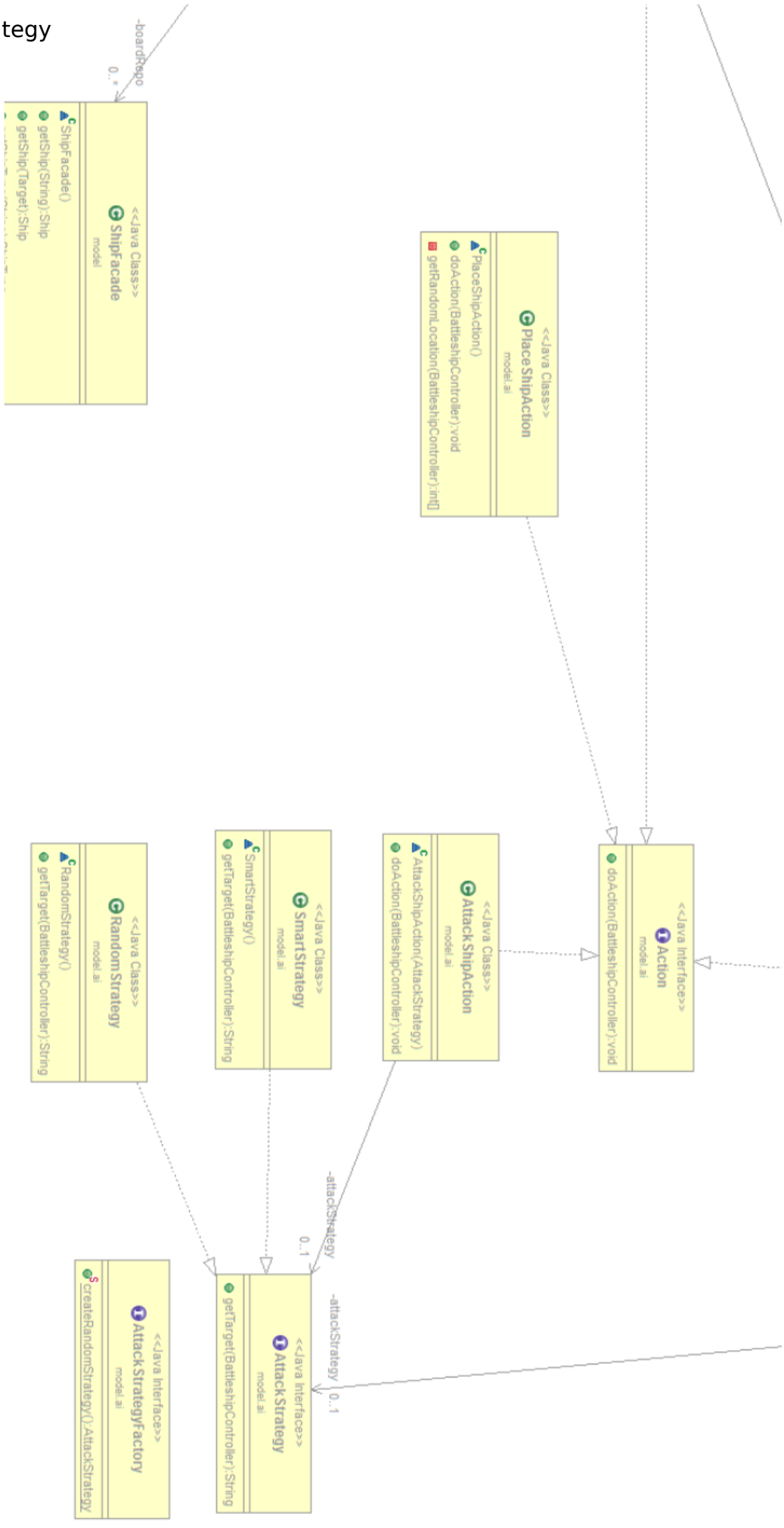
7 ShipRepo


-boardFacade
0..1
BoardFacade
model
BoardFacade()
getShipRepo(String):ShipFacade
<<Java Class>>
-boardRepo
0..*
ShipFacade
model
ShipFacade()
getShip(String):Ship
getShip(Target):Ship
getShipType(String):ShipType
getShipType(Target):ShipType
setShip(Ship):void
getAllShips():List<Ship>
hit(String):Boolean
hit(Target):Boolean
getAvailableShipCount():Map<ShipType,Integer>
-shipRepo
0..1
ShipRepo
model
MaxShips: int
regesterdShips: HashMap<ShipType,Integer>
ShipRepo()
getShip(String):Ship
getShip(Target):Ship
getShipType(String):ShipType
getShipType(Target):ShipType
setShip(Ship):void
getAllShips():List<Ship>
hit(String):Boolean
hit(Target):Boolean
getAvailableShipCount():Map<ShipType,Integer>
<<Java Interface>>
ShipRepoInterface
model
getShip(String):Ship
getShip(Target):Ship
getShipType(String):ShipType
getShipType(Target):ShipType
setShip(Ship):void
getAllShips():List<Ship>
hit(String):Boolean
hit(Target):Boolean
getAvailableShipCount():Map<ShipType,Integer>
<<Java Interface>>
ShipFactory
model.factory
raftcarrier(List<Target>,ShipFacade):Ship
leShip(List<Target>,ShipFacade):Ship
ser(List<Target>,ShipFacade):Ship
troyer(List<Target>,ShipFacade):Ship
marine(List<Target>,ShipFacade):Ship
<<Java Interface>>
hitable
model
getsHit(String):boolean
getsHit(Target):boolean
<<Java Class>>
HealtyState
model.targetState
HealtyState()
damage():TargetState
sink():TargetState
enState
etState
State()
TargetState
etState

8 AIState


BattleShips/facade.ucls
<<Java Class>>
AiFacade
model.ai
AiFacade()
placingDone():void
doAction(BattleshipController):void
<<Java Class>>
PlaceState
model.ai
PlaceState()
getNextAction():Action
<<Java Interface>>
AiStateFactory
model.ai
createAttackState():AiState
createAttackState(AttackStrategy):AiState
createPlaceState():AiState
<<Java Interface>>
Action
model.ai
doAction(BattleshipController):void
<<Java Class>>
MainAi
model.ai
MainAi()
placingDone():void
doAction(BattleshipController):void
<<Java Interface>>
AiState
model.ai
getNextAction():Action
<<Java Class>>
AttackState
model.ai
AttackState(AttackStrategy)
getNextAction():Action

9 Strategy


-boardRego
0..*
ShipFacade()
getShip(String):Ship
getShip(Target):Ship
<<Java Class>>
ShipFacade
model
PlaceShipAction()
doAction(BattleshipController):void
getRandomLocation(BattleshipController):int[]
<<Java Class>>
PlaceShipAction
model.ai
doAction(BattleshipController):void
<<Java Interface>>
Action
model.ai
AttackShipAction(AttackStrategy)
doAction(BattleshipController):void
<<Java Class>>
AttackShipAction
model.ai
SmartStrategy()
getTarget(BattleshipController):String
<<Java Class>>
SmartStrategy
model.ai
RandomStrategy()
getTarget(BattleshipController):String
<<Java Class>>
RandomStrategy
model.ai
-attackStrategy
0..1
getTarget(BattleshipController):String
<<Java Interface>>
AttackStrategy
model.ai
-attackStrategy 0..1
createRandomStrategy():AttackStrategy
<<Java Interface>>
AttackStrategyFactory
model.ai