CS 362 Group Report For Iteration 1

Team 7

Gavin Monroe

Huai Yao Ma

Jonathan Segal

Karthik Subbarao

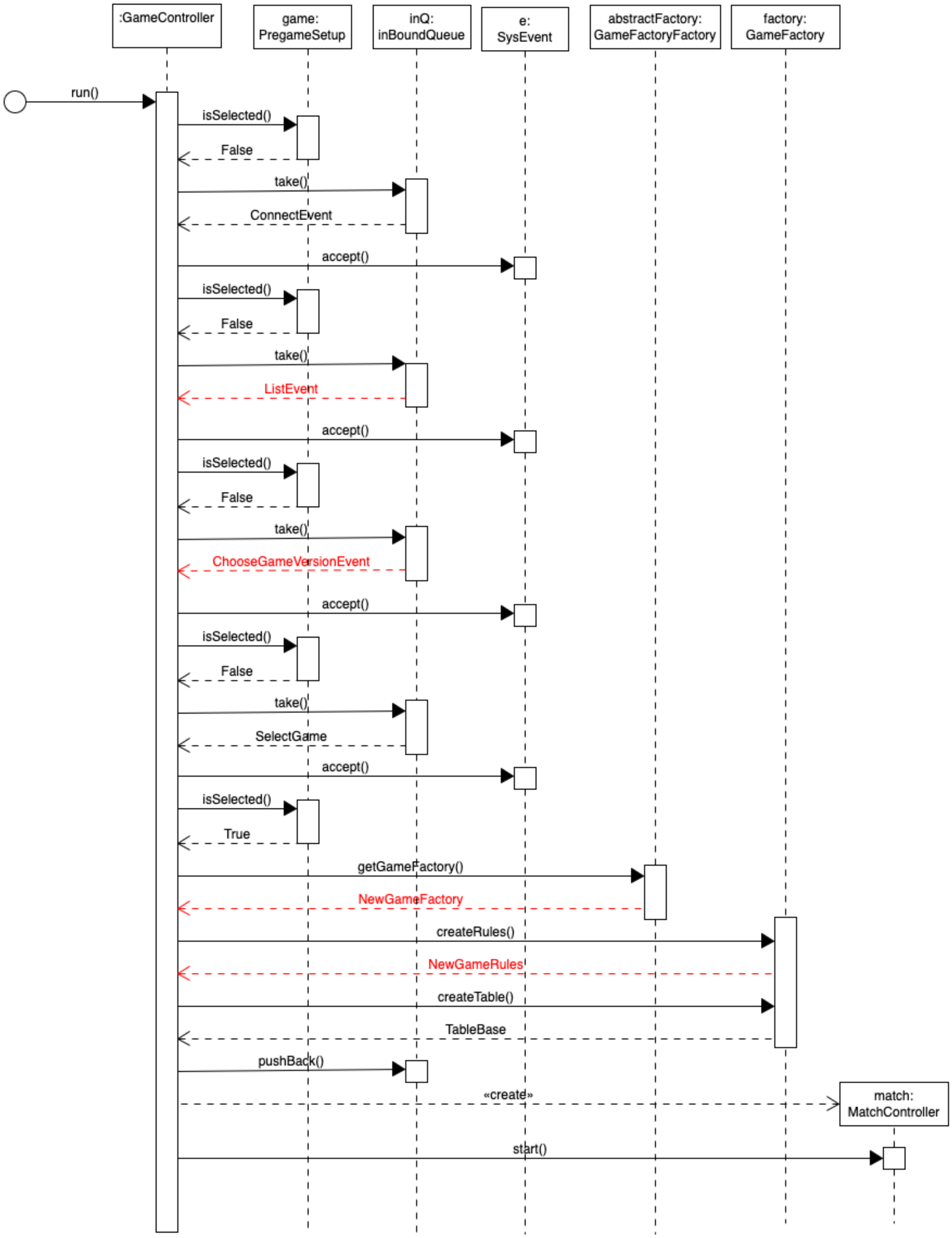
Bo Sheng Lee

For the impact analysis report, our team decided to work on the third feature, which is “Support the host being able to select from multiple rule variations of a game.” First of all, to add this new feature, we need to add some classes to implement this feature. We need to have new events like ListEvent which will display a list of all possible game modes that a user can select, ChooseGameVersionEvent which will be called when a user selects one variation of the game. Then, we also need to add some new packages similar to the fiftytwo packages for each of the new games we want to add. This new package will contain the rules and the specifics for the game that are required to be able to play the new game mode such as the table. Other than that, we also need to modify some classes like GameFactoryFactory to have more versions of games in the list. Method like connect event in the game controller will also need to be modified which now it will not push the SelectGame Event into the queue, instead, it will only push the new ListEvent into the queue. More overload apply() methods for ListEvent and ChooseGameVersionEvent will also need to be added into the GameContoller class.

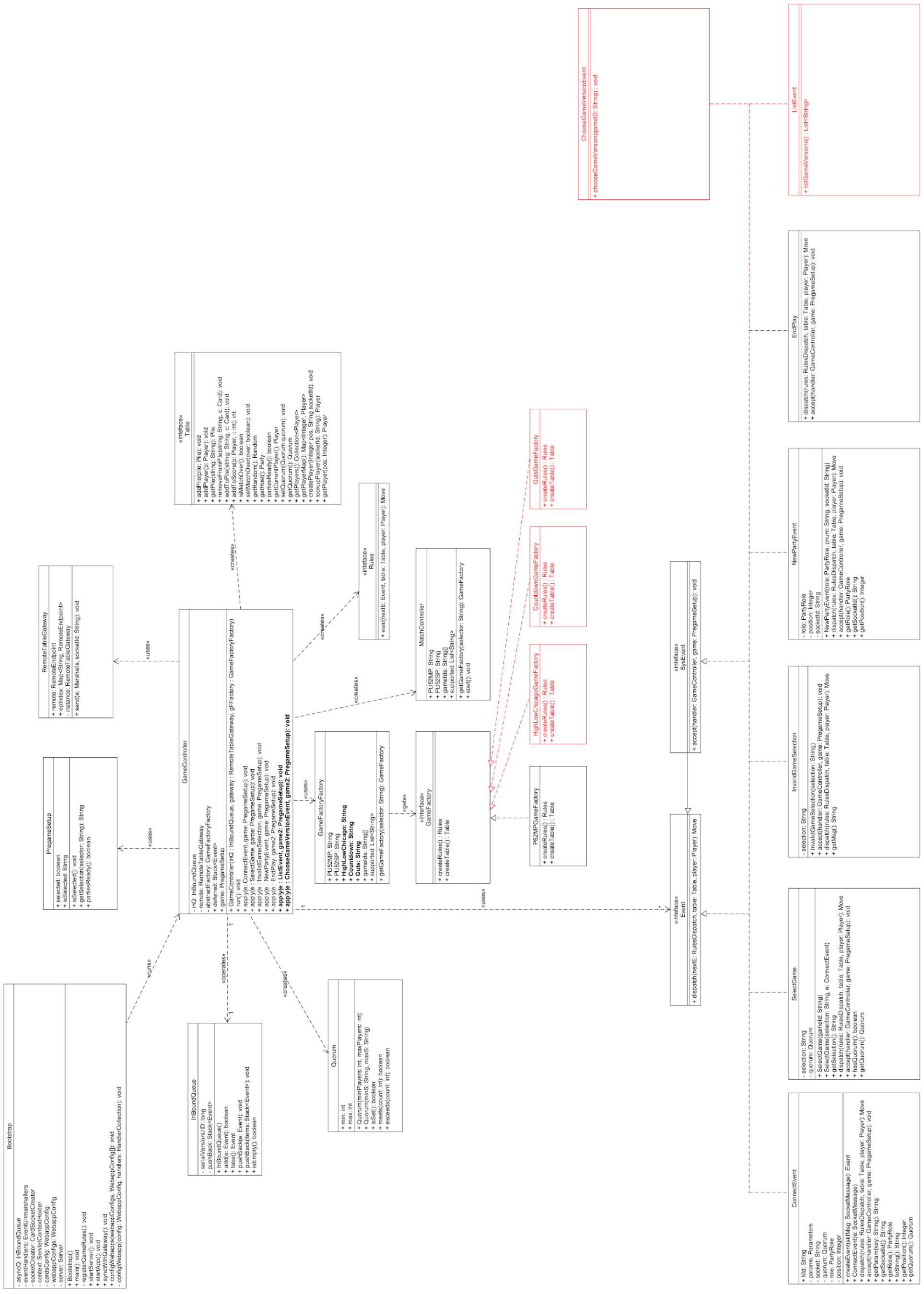
Now, we will go through how the new code will run to support the new feature. We plan to modify the ConnectEvent such that users will use this to connect as a host: 'http://localhost:8080/cards362/?host', and normal players will use this: 'http://localhost:8080/cards362/'. First of all, the ConnectEvent will be taken out from the queue in run() in GameController, and this calls e.accept() which then calls apply(ConnectEvent e, PregameSetup game). Now there are two cases: Host or Normal Player. If the url had a host parameter then inside the apply() with ConnectEvent, it will not push the SelectGame event in the queue, instead, it will push the newly created ListEvent in the queue since the ConnectEvent is created by the host. Next, the ListEvent will be taken from the queue, and apply() will again be called. ListEvent will display a list of different games and wait for user input. We plan to list different game versions (3 games) and player min / max options which will all have redirect urls for example: http://localhost:8080/cards362/?host&player=1&min=2&max=4&game=Guts. When the user selects one game rule, ChooseGameVersionEvent will be called, and apply() will be called as well. In the ChooseGameVersionEvent, the SelectGameEvent will be pushed into the queue. Next, the SelectGameEvent will be taken from the queue, and apply() will be called. It does the same things as what the original SelectGameEvent did. After the game is selected, GameFactoryFactory will produce a GameFactory for the game that the user has selected to play.

Then, it will create rules and tables from the game factory created. After that, the code works the same as the original code before adding the new feature.

Sequence Diagram for GameController



Class Diagram for GameController (Clearer Version on Gitlab folder Iteration1: UpdatedGameControllerClassDiagram.png)


«inteface»
Table
+ addPile(pile: Pile): void
+ addPlayer(p: Player): void
+ getPile(string: String): Pile
+ removeFromPile(string: String, c: Card): void
+ addToPile(string: String, c: Card): void
+ addToScore(p: Player, i: int): int
+ isMatchOver(): boolean
+ setMatchOver(over: boolean): void
+ getRandom(): Random
+ getHost(): Party
+ partiesReady(): boolean
+ getCurrentPlayer(): Player
+ setQuorum(Quorum quorum): void
+ getQuorum(): Quorum
+ getPlayers(): Collection<Player>
+ getPlayerMap(): Map<Integer, Player>
+ createPlayer(Integer pos, String socketId): void
+ lookupPlayer(socketId: String): Player
+ getPlayer(pos: Integer): Player
«creates»
«inteface»
Rules
+ eval(nextE: Event, table: Table, player: Player): Move
«creates»
RemoteTableGateway
+ remote: RemoteEndpoint
+ epIndex: Map<String, RemoteEndpoint>
- instance: RemoteTableGateway
+ send(e: Marshalls, socketId: String): void
«uses»
PregameSetup
+ selected: boolean
+ isSelected: String
+ isSelected(): void
+ getSelection(selector: String): String
+ partiesReady(): boolean
«uses»
GameController
- inQ: InBoundQueue
- remote: RemoteTableGateway
- abstractFactory: GameFactoryFactory
+ deferred: Stack<Event>
+ game: PregameSetup
+ GameController(inQ : InBoundQueue, gateway : RemoteTableGateway, gFFactory : GameFactoryFactory)
+ run(): void
+ apply(e: ConnectEvent, game: PregameSetup): void
+ apply(e : SelectGame, game: PregameSetup): void
+ apply(e : InvalidGameSelection, game: PregameSetup): void
+ apply(e : NewPartyEvent, game: PregameSetup): void
+ apply(e : EndPlay, game2: PregameSetup): void
+ apply(e : ListEvent, game2: PregameSetup): void
+ apply(e : ChooseGameVersionEvent, game2: PregameSetup): void
«creates»
MatchController
+ PU52MP: String
+ PU52SP: String
+ gameIds: String[]
+ supported: List<String>
+ getGameFactory(selector: String): GameFactory
+ start(): void
«uses»
GameFactoryFactory
+ PU52MP: String
+ PU52SP: String
+ HighLowChicago: String
+ Countdown: String
+ Guts: String
+ gameIds: String[]
+ supported: List<String>
+ getGameFactory(selector: String): GameFactory
«gets»
«inteface»
GameFactory
+ createRules() : Rules
+ createTable() : Table
GutsGameFactory
+ createRules() : Rules
+ createTable() : Table
CountdownGameFactory
+ createRules() : Rules
+ createTable() : Table
HighLowChicagoGameFactory
+ createRules() : Rules
+ createTable() : Table
P52MPGameFactory
+ createRules() : Rules
+ createTable() : Table
«uses»
1
*
«runs»
Bootstrap
- asyncQ: InBoundQueue
- eventHandlers: EventUnmarshallers
- socketCreator: CardSocketCreator
- context: ServletContextHolder
- cardsConfig: WebappConfig
- webappConfigs: WebappConfig
- server: Server
+ Bootstrap()
+ main(): void
- registerGameRules(): void
+ startServer(): void
+ startApp(): void
+ syncWithGateway(): void
+ configWebapps(webappConfigs, WebappConfig[]): void
- configWebapp(config: WebappConfig, handlers: HandlerCollection): void
«operates»
1
1
InBoundQueue
- serialVersionUID: long
- pushBack: Stack<Event>
+ InBoundQueue()
+ add(e: Event): boolean
+ take(): Event
+ pushBack(e: Event): void
+ pushBack(items: Stack<Event>): void
+ isEmpty(): boolean
«creates»
Quorum
+ min: int
+ max: int
+ Quorum(minPlayers: int, maxPlayers: int)
+ Quorum(minS: String, maxS: String)
+ isSet(): boolean
+ meets(count: int): boolean
+ exceeds(count: int): boolean
«inteface»
SysEvent
+ accept(handler: GameController, game: PregameSetup): void
«inteface»
Event
+ dispatch(nextE: RulesDispatch, table: Table, player: Player): Move
ChooseGameVersionEvent
+ chooseGameVersion(gameID: String) : void
ListEvent
+ listGameVersions() : List<String>
EndPlay
+ dispatch(rules: RulesDispatch, table: Table, player: Player): Move
+ accept(handler: GameController, game: PregameSetup): void
NewPartyEvent
- role: PartyRole
- position: Integer
- socketId: String
+ NewPartyEvent(role: PartyRole, pnum: String, socketId: String)
+ dispatch(rules: RulesDispatch, table: Table, player: Player): Move
+ accept(handler: GameController, game: PregameSetup): void
+ getRole(): PartyRole
+ getSocketId(): String
+ getPosition(): Integer
InvalidGameSelection
- selection: String
+ InvalidGameSelection(selection: String)
+ accept(handler: GameController, game: PregameSetup): void
+ dispatch(rules: RulesDispatch, table: Table, player: Player): Move
+ getMsg(): String
SelectGame
- selection: String
- quorum: Quorum
+ SelectGame(gameId: String)
+ SelectGame(selection: String, e: ConnectEvent)
+ getSelection(): String
+ dispatch(rules: RulesDispatch, table: Table, player: Player): Move
+ accept(handler: GameController, game: PregameSetup): void
+ hasQuorum(): boolean
+ getQuorum(): Quorum
ConnectEvent
+ kId: String
- params: Parameters
- socket: String
- quorum: Quorum
- role: PartyRole
- position: Integer
+ createEvent(sktMsg: SocketMessage): Event
+ ConnectEvent(e: SocketMessage)
+ dispatch(rules: RulesDispatch, table: Table, player: Player): Move
+ accept(handler: GameController, game: PregameSetup): void
+ getParam(key: String): String
+ getSocketId(): String
+ getRole(): PartyRole
+ toString(): String
+ getPosition(): Integer
+ getQuorum(): Quorum