COMMUNICATION PROTOCOL

[player] {“nickname”: nickname} ->

X <- {“status”: “fatalError”, “message”: “Lobby is already full”} **[server]**

X <- {“status”: “fatalError”, “message”: “Nickname has been taken”} **[server]**

X <- {“status”: “fatalError”, “message”: “The nickname is not valid”} **[server]**

X <- {“status”: “fatalError”, “message”: “A game already started”} **[server]**

X <- {“status”: “fatalError”, “message”: “The game already ended”} **[server]**

broadcast <- {“status”: “ok”, “type”: “playerList”, “message”: {“playerList”: [{“nickname”: “player1”, “isHost”: true}, {“nickname”: “player2”, “isHost”: false}]}} **[server]**

[host] {“command”: “startGame”} ->

[host] {“command”: “startGame”, “config”: config.json} ->

X <- {“status”: “error”, “message”: “Not the host”} **[server]**

X <- {“status”: “error”, “message”: “Invalid config file”} **[server]**

broadcast <- {“status”: “ok”, “type”: “config”, “message”: {“config”: config.json, “turnOrder”: String[]}} **[server]**

broadcast <- {“status”: “ok”, “type”: “update”, “message”: {game state}} **[server]**

broadcast <- {“status”: “ok”, “type”: “command”, “message”: {“player”: String, “value”: CommandBuffer}} **[server]**

[player] {
    “command”: “initDiscard”,
    “value”: int[] // indices of the cards to discard
}

broadcast <- {“status”: “ok”, “type”: “update”, “message”: {game update}} **[server]**

broadcast <- {“status”: “ok”, “type”: “command”, “message”: {“player”: String, “value”: CommandBuffer}} **[server]**

[player] {
    “command”: “initResources”,
    “value”: ConcreteResourceSet[] // a ConcreteResourceSet for each depot
}

broadcast <- {“status”: “ok”, “type”: “update”, “message”: {game update}} **[server]**

broadcast <- {“status”: “ok”, “type”: “command”, “message”: {“player”: String, “value”: CommandBuffer}} **[server]** game state:

```
{
    “gameZone”: {
        “marbleMarket”: {
            “market”: MarbleColour[][],
            “freeMarble”: MarbleColour
        },
        “cardMarket”: {
            “market”: {“topCard”: Integer (null if empty), “quantity”: int}[][]
        },
        “actionTokenDeck”: { // only in single player
            “flippedActionToken”: ActionToken
        }
    },
    “players”: [
        {
            “nickname”: String.
            “inkwell”: boolean,
            “board”: {
                “faithTrack”: {
                    “position”: int,
                    “receivedFavors”: int[] // array of received pope favors ids
                },
                “warehouse”: {
                    “depots”: ConcreteResourceSet[]
                },
                “strongbox”: {
                    “content”: ConcreteResourceSet
                },
                “devCards”: {
                    “decks”: [
                        {devCardDeck}, -> Integer[] // array of dev cards ids
                        {devCardDeck},
                        {devCardDeck}
                    ]
                },
                “playedLeaderCards”: {
                    “leaderCards” : Integer[] // array of leader cards ids
                },
                “handLeaderCards”: {
                    “leaderCards”: Integer[] // array of leader cards ids, null if hidden
                }
            }
        },
        {
            “nickname”: String,
            “inkwell”: boolean,
            …
        }
    ]
}
```

Turns

```
[player] {
        “request”: “newCommand”,
        “command”: CommandType
}
```

broadcast <- {“status”: “ok”, “type”: “command”, “message”: {“player”: String, “value”: CommandBuffer}} **[server]**

```
[player] {
        “request”: “action”,
        “command”: String,
        “value”: JsonElement
}
```

```
[player] {
        “request”: “cancel”
}
```

CommandTypes:

- InitDiscard
  actions:
  - command: “indices”
    value: int[]
- InitResources
  actions:

  - command: “resources”
    value: ConcreteResourceSet[]
- DiscardLeader
  actions:
  - command: “index”
    value: int
- PlayLeader
  actions:
  - command: “index”
    value: int
- Market
  actions:
  - command: “selection”
    value: {
        “rowColSel”: boolean,
        “index”: int
    }
  - command: “conversion”
    value: ConcreteResource[]
  - command: “addToDepot”
    value: {
        “depotIndex”: int,
        “resources”: ConcreteResourceSet
    }
  - command: “removeFromDepot”
    value: {
        “depotIndex”: int,

“resources”: ConcreteResourceSet
}

  - command: “swapFromDepots”
    value: {
    “depotIndexA”: int,
    “depotIndexB”: int
    }
  - command: “confirmWarehouse”
    value: null
- Purchase
  actions:
  - command: “cardSelection”
    value: {
    “row”: int
    “column”: int
    }
  - command: “deckSelection”
    value: int
  - command: “spendResources”
    value: {
    “warehouse”: ConcreteResourceSet[],
    “strongbox”: ConcreteResourceSet
    }
- Production
  actions:
  - command: “selection”:
    value: int[]

- ○ command: “spendResources”
  value: {
  “warehouse”: ConcreteResourceSet[],
  “strongbox”: ConcreteResourceSet
  }
- ○ command: “choiceSelection”
  value: ConcreteResource[]

broadcast <- {“status”: “ok”, “type”: “update”, “message”: {game update}} **[server]**

broadcast <- {“status”: “ok”, “type”: “command”, “message”: {“player”: String, “value”: CommandBuffer}} **[server]**

[player] {
“request”: “endTurn”
}

{“status”: “ok”, “type”: “endGame”, “value”: EndGameMessage} **[server]**

EndGameMessage:

```
{
        “results”: [
                {
                        “player”: String,
                        “basePoints”: int,
                        “resources: int
                },
                {
                        “player”: String,
                        “basePoints”: int,
                        “resources: int
                },
                ...
        ],
        “computerWin”: boolean // only in single player
}
```