Configuration Template :

The configuration is built in JSON format (start with "{" and ends with "}").

The first atribute inside the brackets is “Type”.
According to the configuration type, all of the other atributes in the configuration are determined.

```
"Type"          : Type of the game. [String]
                  Can be one of the following:
                  “createConfig”, “runConfig”, agent”.
```

If type is “createConfig” :

In order to create configuration file and to upload it to the server.
The structure of the configuration will be this way:

```
{
        "Type"          : createConfig,
        "Global"        : # Will Be Explained,
        "Games"         : # Will Be Explained
}
```

```
        "Global"        : An object who gives an overall description of the game
                          and has the folowing structure
                {
                        "ID"                    : # Will Be Explained,
                        "RESEARCHER_NAME        : # Will Be Explained,
                        "Colors"                : # Will Be Explained,
                        "boards"                : # Will Be Explained
                }

         "Games"        : An Array of objects who specify exactly how each game is
                           Conducted. each game has the following structure:

                {
                        "GAME_NAME"             : # Will Be Explained,
                        "Board"                 : # Will Be Explained,
                        “AutomaticChipSwitch”   : # Will Be Explained,
                        "roles"                 : # Will Be Explained,
                        "phases"                : # Will Be Explained,
                        "rounds"                : # Will Be Explained,
                        “gameConditions”        : # Will Be Explained,
                        “players”               : # Will Be Explained,
                        “agents”                : # Will Be Explained
                }
```

“games” – Structure explenation.

Specification of “Global” attributes:

“ID” [int] - An ID number given to the configuration
“RESREARCHER_NAME” [string] – The name of the researcher.
“Colors” [Array of strings] - The colors of which the board will be built. Can be each of the following:
“pink”, “blue”, “purple”, “green”, “yellow”, “darkblue”.
“boards” – specify the boards that will be used in the game and has the following structure:

```
{
        “board_1”      : # Will Be Explained,
        “board_2”      :
        .
        .
        .
}
```

“board_1” explenation:
Each board specified in “boards” is a N*M Array, each cell in the array is [int] between 0 to **[“Colors” array size] – 1.**

Specification of “Games” attributes:

“GAME_NAME” [string] – the game’s name.
“Board” [string] – a name of a board on which the players will move (must be one of the boards defined in “Global”) .
“AutomaticChipSwitch” [0 or 1] – specify whether chips will be sent automatically when an offer is accepted.
“AutoCounterOffer” – [optional], when a player rejects an offer imidiatly a counter offer row is oppened for him to respond.
“roles” – defines the roles each player can have. And have the following structure :

```
{
        “role_1” : { # Will Be Explained }
        .
        .
        .
}
```

Roles explenation
Each role can have the following fields :

canMove - [0 or 1]
canOffer - [0 or 1]
canTransfer - [0 or 1]
canSeeChips - [0 or 1]
canSeeLocations - [0 or 1]
num_of_offers_per_player – how manny offers a player with
    this role can send to each other player in each phase.
total_num_of_offers – the maximum amount of offers can be
    sent in total to the other players.
canOfferTo – an array of **roles,** specify to which roles a player
    with this role can send offers.

- Note:
    - The default for the first 5 is **0**.
    - The default for the rest is no limit.

```
“phases” : describes the phases in the game (each round will be
        composed from these phases), and will have the following
        structure:
{
        “phase1” : { # Will Be Explained }
        .
        .
        .
}
```

```
Each phase will have the following structure:
{
        “name”              : # Will Be Explained,
        “time”              : # Will Be Explained,
        “players_roles”     : # Will Be Explained,
}
```

Phase explenation:
“name” [string] – name of the phase.
“time” [int] – duration of the phase in **miliseconds.**
“players_roles” – althogh there’s a role for each player, The researcher
    can overide the players’ roles’ actions.
    “players_role” is an array of bindings [player, actions] and
    each binding has the following structure:

```
{
        “id”                    : [id of the player]
        “additional_actions”    : # As Explained Above.
}
```

"rounds" : describes the rounds in the game, and has the following structure:

```
{
        "General"               : # Will Be Explained,
        "rounds_definitions     : # Will Be Explained,
}
```

"General" explenation:

Has one attribute –
numberOfTimesToRepeatRounds – defines the amount of times to repeat each of the rounds. Default is 1.

"rounds_definition" explenation:

An array rounds definitions. Each round has the folowing structure:

```
{
        "name"                  : # Will Be Explained,
        "phases_in_round"       : # Will Be Explained,
        "players_roles"         : # Will Be Explained,
}
```

"name" [string] – name of the round.
"phases_in_round" – array of phases names (there must be phases with the mentioned names in the phases list)

"players_roles" – an array of bindings for each player what's his role.

Each binding has the following structure:

```
{
        "id"    : [one of the players ID],
        "role"  : [one of the roles defined above]
}
```

"GameConditions" – describes the goal cordinates and scoring methods, has the following structure:

```
{
        "GoalCordinates"        : # Will Be Explained,
        "gameGoal"              : # Will Be Explained,
        "endConditions"         : # Will Be Explained,
        "score"                 : # Will Be Explained,
}
```

“GoalCordinates” – An array of points each point Pi has the following Structure: [Xi,Yi].
“gameGoal” [string] – can be “max_points” or “min_points”
“endCondition” [object] – has the following structure:

```
{
        “numOfRoundsStandStill” : [int]
}
```

- The game will end after the amount of rounds mentioned above in which the game hasn’t changed.

“score” – describes the scoring function used to calculate players’ scores. Has the following structure:

```
{
        “onReachGoalGoalView”      : [int]
        “onReachGoalPlayerView”    : [int]
        “pointsPerChips”           : [int]
}
```

onReachGoalGoalView	– a player who is also a goal get point for each other player who reached him.
onReachGoalPlayerView	– points for reaching a goal on the board.
pointPerChip	– points for each chip in the stash.

“players” – An array of the players participating in the game. Each player has the following structure:

```
{
        “id”          : [int – player’s id].
        “name”        : [string – player’s name].
        “basic_role”  : [string – one of the roles defined above].
        “locationX    : [int – X cordination on the board].
        “locationY”   : [int – Y cordination on the board].
        “chips”       : [array of integers, how manny chips will the
                        player have of each of the colors mentioned
                        above, accordingly].
        “Goals”       : #Will Be Explained.
        “isGoal”      :[0 or 1 – mentions if the player is goal].
}
```

“Goals” Explenation:
An array of goals – each of the goals has the following structure:

```
{
        “type”        : [string – can be : “plain” or “player”]
        “x”           : [Integer – x value of the goal].
        “y”           : [Integer – y value of the goal].
        “real”        : [0 or 1   – is the goal real or fake].
        “isShown”     : [0 or 1   – is the goal to be shown or not].

}
```

“agents” – An array of the agents participating in the game.
Each agent has the same structure exactly as in “players”.

If type is “runConfig” :

[Use in order to run configuration file/s and by that start games.]

The structure of the configuration will be this way:

```
{
        "Type"                  : runConfig,
        “confsToRun”            : # Will Be Explained
}
```

“confsToRun” – an Array of objects, each object has the following structure:

```
{
        “confID”        : [ int – the configuration’s ID ],
        “playerList”    : # Will Be Explained,
        “agentList”     : # Will Be Explained,
}
```

“playerList” [array of integers] – the IDs of the human players who participate in the games. (defined in the configuration file)
“agentList” [array of integers] – the IDs of the agents who participate in the games. (defined in the configuration file)

- Each of the objects in the “confsToRun” array is describes the way to run games between players.
  The players in the first structure in the array will play **all** the games in the configuration file mentioned in the first structure.
  After they finish playing the games, the games in the next struxture will start etc.

If type is “Agent” :

[Use in order to send messages from agent to the server]

There are few possible messages who can be sent to the server.

Joining a game:

```
{
        "Type"              : "Agent",
        "Action"            : "joinGame",
        "ID"                : [ int – the agent ID]
        "listening_port"    : [ int - the port the agent listening]
        "IP"                : [ string - the IP address of the agent]
}
```

Moving on board:

```
{
        "Type"              : "Agent",
        "Action"            : "moveUp"/"moveRight"/"moveLeft"/"moveDown"
        "ID"                : [ int – the agent ID given when joinGame
                                                    message sent],
        "gameId"            : [int – the game ID]
}
```

Transfering data

```
{
        "Type"                   : "Agent",
        "Action"                 : "transferData",
        "ID"                     [ int – the agent ID given when joinGame
                                                    message sent],
        "gameId"                 : [int – the game ID],
        "JcolorsToSend"          : [ Arrary of integers – chips you want to send],
        "recieverId"             : [int – receiver ID defined in the configuration],
        "sentFrom"               : [ int – the agent ID],
        "transferId"             : [int – transaction ID],

}
```

Sending an offer to another player

```
{
        "Type"              : "Agent",
        "Action"            : "sendOffer",
        "ID"                : [ int – the agent ID given when joinGame
                                                    message sent],
        "gameId"            : [int – the game ID],
        "JcolorsToGet"      : [ Arrary of integers – chips you want to get ],
        "JcolorsToOffer"    : [ Arrary of integers – chips you want to send],
        "recieverId"        : [ int – receiver ID ],
        "sentFrom"          : [ int – the agent ID defined in the configuration],
        "offerId"           : [ int – offer’s ID ]
}
```

Reject an offer

```
{
        "Type"              : "Agent",
        "Action"            : "rejectOffer",
        "ID"                [ int – the agent ID given when joinGame
                                              message sent],
        "sentFrom"          : [ int – ID of sender ],
        "offerId"           : [ int – offer’s Id] ,
        "gameId"            : [ int – game‘s Id ]
}
```

Accept an offer:

```
{
        "Type"              : "Agent",
        "Action"            : "acceptOffer",
        "ID"                : [ int – the agent ID given when joinGame
                                              message sent],
        "player1"           : [player one’s name],
        "player2"           : [player two’s name],
        "gameId"            : [int – game’s Id        ]
}
```