

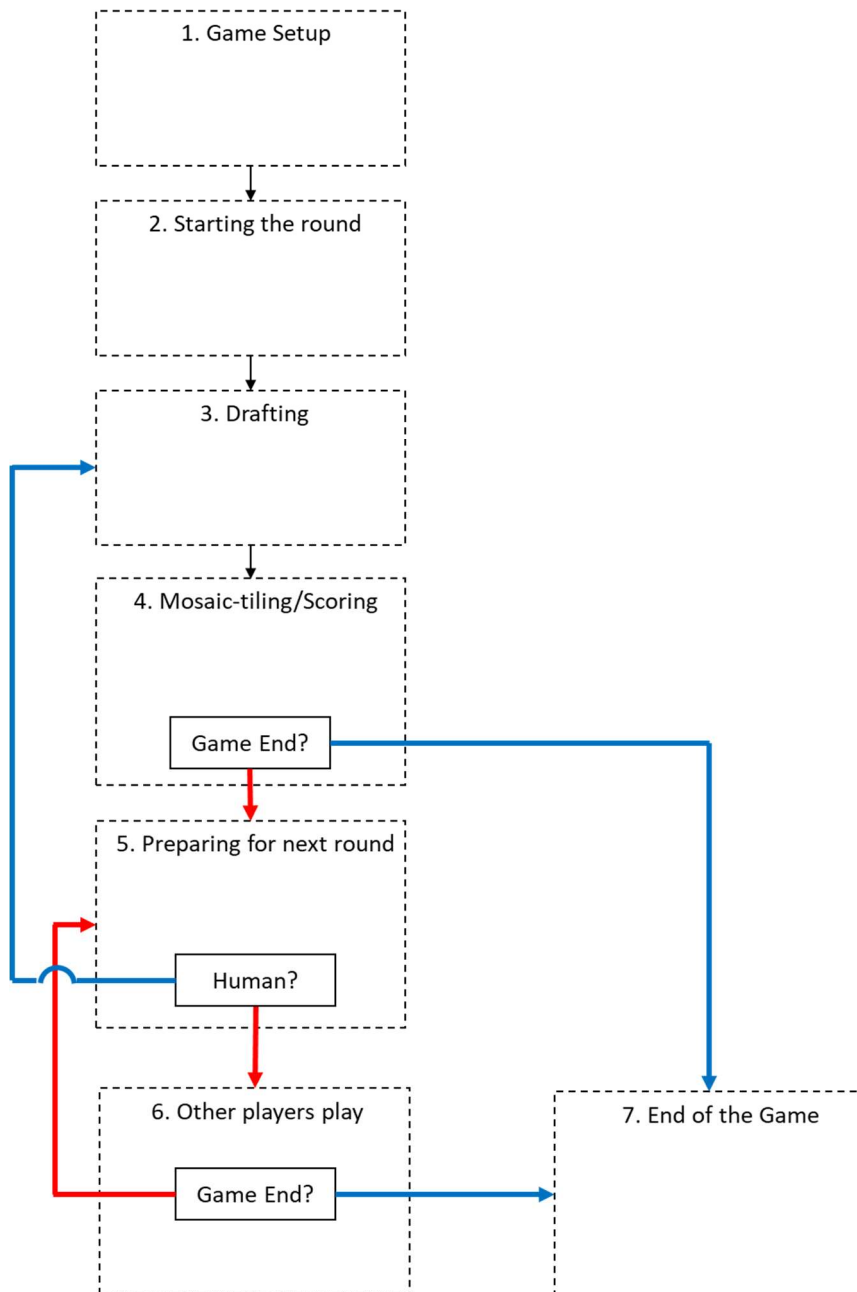
# Improved Skeleton of Azul Game

## (Assignment 2, D2C)

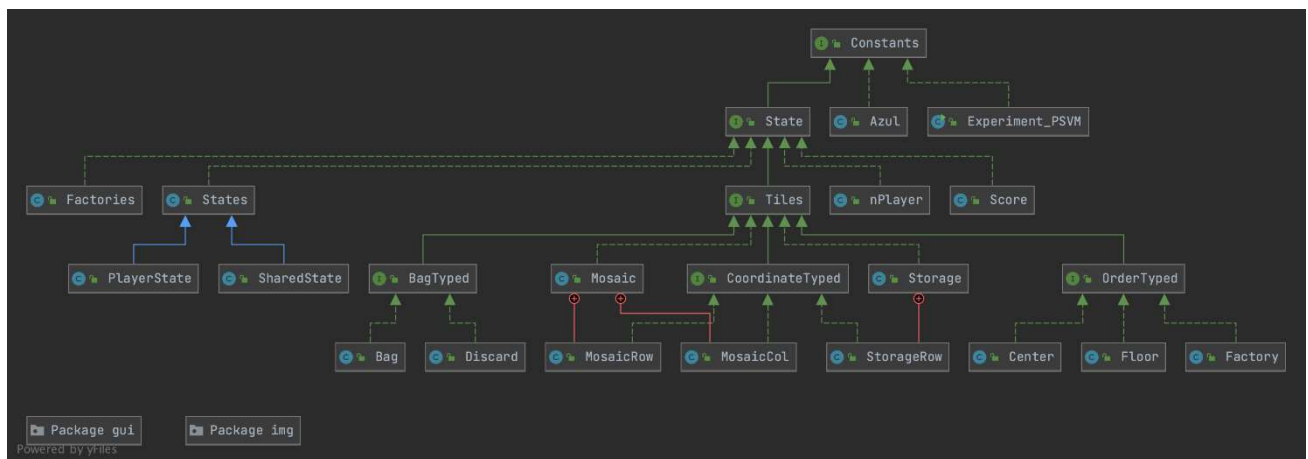
Author 1: John (Min Jae), Kim (u7269158)

Author 2: Si bo, Hu (u7271125)

### 1. Brief logic flow of Azul skeleton (block diagram)

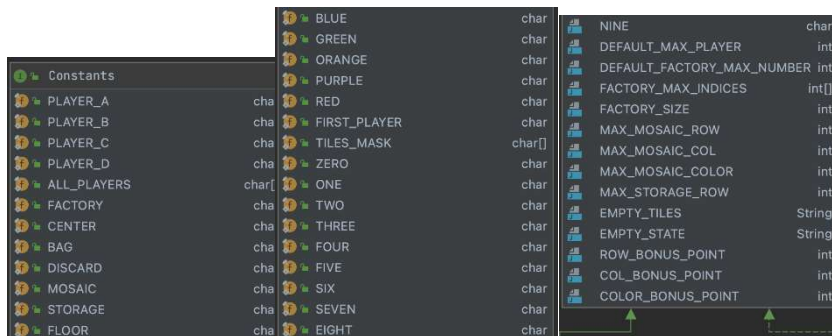


## 2. Brief structure of Azul skeleton



### A. Interfaces

A-a. Constants: every important constant is implemented from here



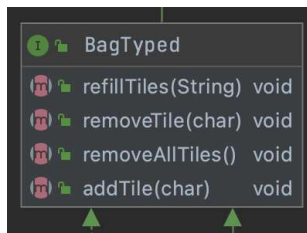
A-b. State: every method related to strings of state.



A-c. Tiles: every method related to counting tiles.



A-d. BagTyped: every method related to class which stores tiles as a form of “NNNNNNNNNNNN”



A-e. OrderedTyped: every method related to class which stores tiles as a form of “alphabetical order”

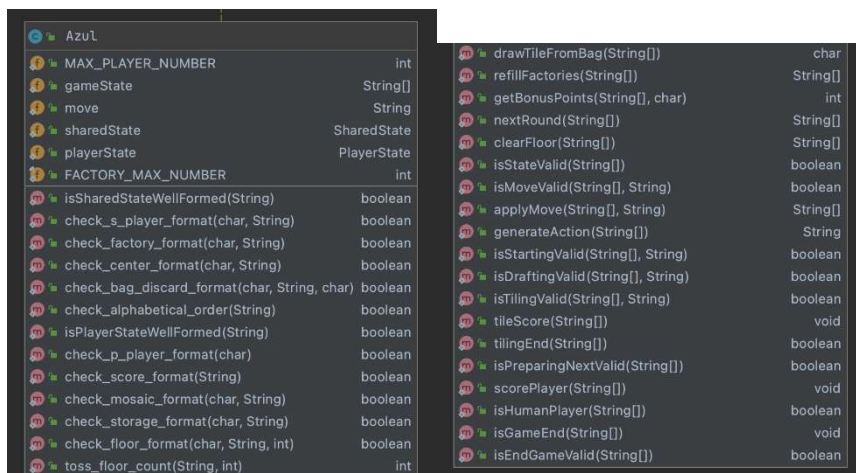


A-f. CoordinateTyped: every method related to class which stores tiles as a form of alphabet with coordinate, or row and number of tiles

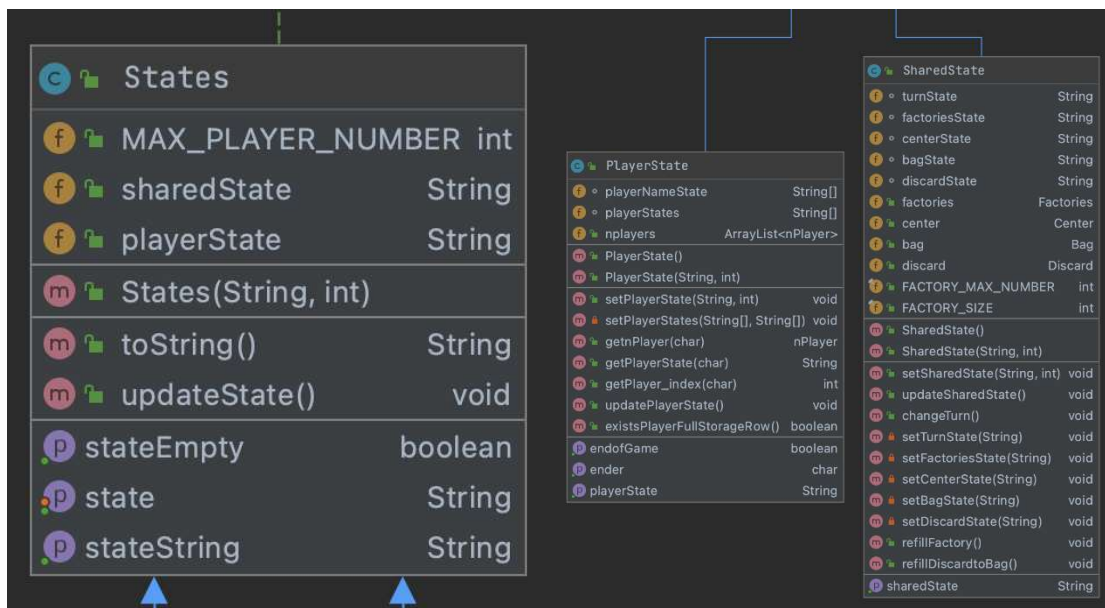


## B. Classes and inner classes

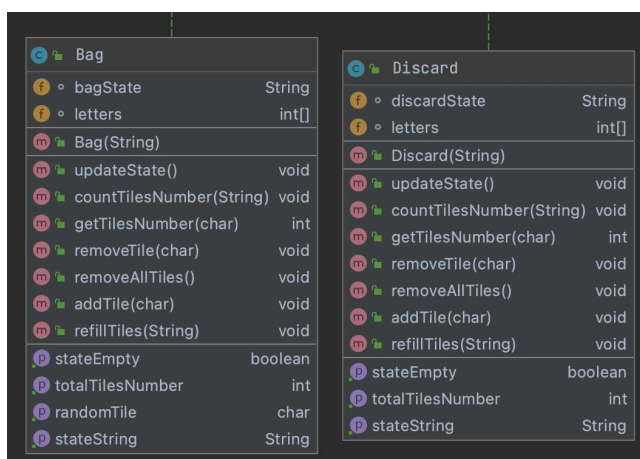
B-a. Azul : main controlling class of game flow, communicates with package gui related class Game, Viewer



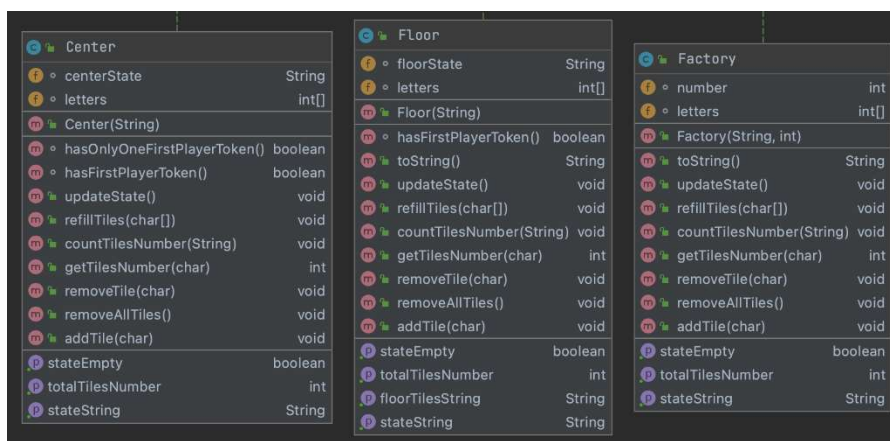
B-b. State, SharedState, PlayerState: shared, playerstate extends state class. They store state string information and divide them to make smaller classes



B-c. Bag, Discard: classes bag and discard implement BagTyped interface



B-d. Center, Floor, Factory: classes center, floor and factory implements OrderTyped interface.



B-e. Mosaic, Storage: classes Mosaic, Storage implements Tiles interface and hav inner classes MosaicRow, MosaicCol, StorageRow which implements CoordinatiedTyped interface,












<b>Mosaic</b> <ul style="list-style-type: none"> <li>◦ mosaic_rows ArrayList&lt;MosaicRow&gt;</li> <li>◦ mosaic_cols ArrayList&lt;MosaicCol&gt;</li> <li>◦ letters int[]</li> </ul> <ul style="list-style-type: none"> <li>➤ Mosaic(String)</li> <li>➤ addMosaicRowCol(String) void</li> <li>➤ getMosaicRow(int) MosaicRow</li> <li>➤ getMosaicCol(int) MosaicCol</li> <li>➤ countTilesNumber(String) void</li> <li>➤ getTilesNumber(char) int</li> <li>➤ toString() String</li> <li>➤ updateState() void</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ totalTilesNumber int</li> <li>Ⓟ stateString String</li> </ul>	<b>MosaicRow</b> <ul style="list-style-type: none"> <li>◦ MOSAIC_MASK char[]</li> <li>◦ letters int[]</li> <li>◦ MAX_TILES_LIMIT int</li> <li>◦ row int</li> </ul> <ul style="list-style-type: none"> <li>➤ MosaicRow(String, int)</li> <li>➤ generateMosaicMask(int) void</li> <li>➤ countTilesNumber(String) void</li> <li>➤ getTilesNumber(char) int</li> <li>➤ compareTo(MosaicRow) int</li> <li>➤ toString() String</li> <li>➤ updateState() void</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ totalTilesNumber int</li> <li>Ⓟ rowTilesFull boolean</li> <li>Ⓟ stateString String</li> </ul>	<b>MosaicCol</b> <ul style="list-style-type: none"> <li>◦ MOSAIC_MASK char[]</li> <li>◦ letters int[]</li> <li>◦ MAX_TILES_LIMIT int</li> <li>◦ col int</li> </ul> <ul style="list-style-type: none"> <li>➤ MosaicCol(String, int)</li> <li>➤ generateMosaicMask(int) void</li> <li>➤ countTilesNumber(String) void</li> <li>➤ getTilesNumber(char) int</li> <li>➤ compareTo(MosaicCol) int</li> <li>➤ toString() String</li> <li>➤ updateState() void</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ totalTilesNumber int</li> <li>Ⓟ rowTilesFull boolean</li> <li>Ⓟ stateString String</li> </ul>
--	---	---

<b>Storage</b> <ul style="list-style-type: none"> <li>◦ storage_rows ArrayList&lt;StorageRow&gt;</li> <li>◦ letters int[]</li> </ul> <ul style="list-style-type: none"> <li>➤ Storage(String)</li> <li>➤ addStorageRow(String) void</li> <li>➤ countTilesNumber(String) void</li> <li>➤ getTilesNumber(char) int</li> <li>➤ isStorageTilesValid() boolean</li> <li>◦ existsStorageRowTilesFull() boolean</li> <li>◦ isStorageRowTilesFull(int) boolean</li> <li>➤ getStorageRow(int) StorageRow</li> <li>➤ toString() String</li> <li>➤ updateState() void</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ totalTilesNumber int</li> <li>Ⓟ stateString String</li> </ul>	<b>StorageRow</b> <ul style="list-style-type: none"> <li>◦ storage_row_color char</li> <li>◦ row int</li> <li>◦ MAX_TILES_LIMIT int</li> <li>◦ letters int[]</li> </ul> <ul style="list-style-type: none"> <li>➤ StorageRow(String, int)</li> <li>◦ isStorageRowTileColorValid(char) boolean</li> <li>◦ isStorageRowTilesValid() boolean</li> <li>➤ removeTile(char) void</li> <li>➤ removeAllTiles() void</li> <li>➤ addTile(char) void</li> <li>➤ compareTo(StorageRow) int</li> <li>➤ toString() String</li> <li>➤ updateState() void</li> <li>➤ countTilesNumber(String) void</li> <li>➤ getTilesNumber(char) int</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ totalTilesNumber int</li> <li>Ⓟ rowTilesFull boolean</li> <li>Ⓟ stateString String</li> <li>Ⓟ tilesColor char</li> </ul>
---	---

B-f. Factories, nPlayer: classes Factories stores ArrayList of Factory, PlayerState stores ArrayList of nPlayer. Both implements State interface.

<b>Factories</b> <ul style="list-style-type: none"> <li>◦ factoriesState String</li> <li>◦ max_factories_number int</li> <li>◦ factory ArrayList&lt;Factory&gt;</li> </ul> <ul style="list-style-type: none"> <li>➤ Factories(String, int)</li> <li>➤ getFactoryTilesNumber(char) int</li> <li>➤ getFactory(int) Factory</li> <li>➤ toString() String</li> <li>➤ updateState() void</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ factoryTotalTiles int</li> <li>Ⓟ stateString String</li> <li>Ⓟ factories String</li> <li>Ⓟ factoryFull boolean</li> </ul>	<b>nPlayer</b> <ul style="list-style-type: none"> <li>◦ nplayerState String</li> <li>◦ scoreState String</li> <li>◦ mosaicState String</li> <li>◦ storageState String</li> <li>◦ floorState String</li> <li>◦ score Score</li> <li>◦ mosaic Mosaic</li> <li>◦ storage Storage</li> <li>◦ floor Floor</li> </ul> <ul style="list-style-type: none"> <li>➤ nPlayer(String, char)</li> <li>➤ setScoreState(String) void</li> <li>➤ setMosaicState(String) void</li> <li>➤ setStorageState(String) void</li> <li>➤ setFloorState(String) void</li> <li>◦ existsStorageRowTilesFull() boolean</li> <li>➤ updateState() void</li> </ul> <ul style="list-style-type: none"> <li>Ⓟ stateEmpty boolean</li> <li>Ⓟ ender boolean</li> <li>Ⓟ nPlayerChar char</li> <li>Ⓟ stateString String</li> <li>Ⓟ nPlayerState String</li> </ul>
--	---

B-g. Score: classes Score calculates score of each player.

	Score	
	EMPTY_STATE	String
	◦ score	int
	Score(String)	
	addScore(int)	void
	scoreLose_clearFloor(int)	int
	clearFloorScore(int)	void
	updateState()	void
	stateEmpty	boolean
	scoreState	String
	stateString	String