OBJECT-ORIENTED PROGRAMMING HY-252

AMPHIPOLIS

by Dimitris Grammenidis, csd3933

Phase B

Content Table:

ENJOY!

MVC Model

This project will follow the MVC model - thus the core will be separated from the GUI, and everything will be coordinated through the Controller.

Packages

-Controller package - features the Controller class.
-Model package - features the different packages related to models
-View package - features the GUI, Menu and Playernum classes.

The Model package features the following packages:
-boardPack, which features the Board class
-characterPack, which features the Characters abstract class as well as its subclasses Archaeologist, Assistant, Digger and Professor
-playerPack, which features the Player class
-tilePack, which features the Bag class, the Tile abstract class as well as its subclasses - class LandslideTile and abstract class FindingTile, which has the subclasses - class AmphoraTile, class MosaicTile, class SkeletonTile and abstract class StatueTile, which has subclasses SphinxTile and CaryatidTile.

We will go in depth about all those in this report and also show the respective UML diagrams.

Board pack

MODEL PACK

The Board.java class is the board of the game, containing the several areas of the board which are ready to be filled with their respective tiles, and some methods which help with initializations.
entryArea, statueArea, mosaicArea, skeletonArea and amphoraArea are the ArrayLists of the specific size and tile type that will represent the five areas of the board.

Method Details

drawTile

```
public Tile drawTile(Bag bag)
```

Transformer(Mutative): Draws a tile from the bag
Preconditions: Every turn
Postconditions: Draws a tile from the bag
Invariants: -

setFirst

```
public int setFirst(Player[] players)
```

Transformer(Mutative): Sets the first player based on last museum visit
Preconditions: beginning of game after players are created
Postconditions: Sets the first player based on last museum visit
Invariants: -

checkGameOver

```
public boolean checkGameOver()
```

Accessor: Returns game status (true if over, false if not over)
Preconditions: -
Postconditions: Returns game status
Invariants: -

makePlayersAndCharacters

```
public Player[] makePlayersAndCharacters()
```

Transformer(Mutative): Initializes players and characters
Preconditions: beginning of game
Postconditions: Initializes necessary components
Invariants: -

Phase 2 comments:

As you may notice, i have removed most of the functions from Board (added only the makePlayersAndCharacters here), and implemented their function in Controller, as i found it more convienient.

CharacterPack

Basically the class that matters is Character. The rest are there for hierarchy - modularity purposes and do not serve any practical reason for my outlook of the current project.

Character:

-private boolean hasBeenUsed: is a boolean variable for determining if a character has been used or not.

-private String belongsTo: is a String that contains the name of the player who this character belongs to

-private charColor color: is an object of the enumeration charColor that shows the color this character has

-public enum charColor: limits the acceptable colors to the specified ones in the enumeration

-public enum chars: limits characters taken as arguments to the ones in the enumeration

Method Details

gethasBeenUsed	getColor
`public boolean gethasBeenUsed()` **Accessor:** Returns if character has been used or not (part of set-get methodology) **Preconditions:** - **Postconditions:** Returns if character has been used or not (part of set-get methodology) **Invariants:** -	`public Characters.charColor getColor()` **Accessor:** Returns character color (part of set-get methodology) **Preconditions:** - **Postconditions:** Returns character color(part of set-get methodology) **Invariants:** -

Phase 2 comments:

I removed the getBelongsTo and setBelongsTo as they were redundant for my implementation (characters belong to a specific player and never change who they belong to)

CharacterPack

setColor

```
public void setColor(Characters.charColor color)
```

Transformer(Mutative): Sets color of character (part of set-get methodology)
Preconditions: character is assigned to a player of same color
Postconditions: Sets color of character (part of set-get methodology)
Invariants: -

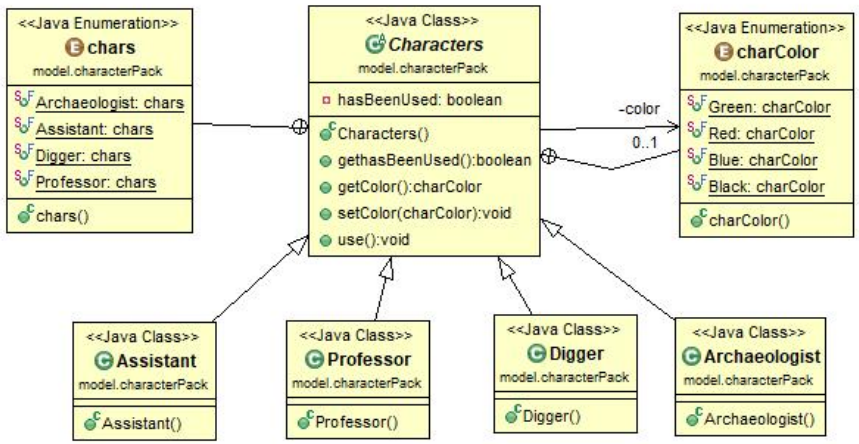
use

```
public void use()
```

Transformer(Mutative): Sets hasBeenUsed to 1 (part of set-get methodology)
Preconditions: character is not used (part of set-get methodology)
Postconditions: Sets hasBeenUsed to 1 (part of set-get methodology)
Invariants: -

I am not planning to override use inside each child class. Instead, the indirect implementation will happen inside the Controller package as it is more convenient.

UML Diagram



PlayerPack

The PlayerPack contains only the Player class, which has:
-private String name is the name of the Player
-private int lastMuseumVisit is the number of days since the last visited museum, used for calculating who plays first
-private charColor color is from enum charColor - players use the same colors so it is a convient setup
-private ArrayList<FindingTile> playerTiles is an ArrayList with all of the player's tiles
-private ArrayList<Characters> playerCharacters is an ArrayList with all of the player's characters

Method Details

getName

```
public java.lang.String getName()
```

Accessor: Returns name of player (part of set-get methodology)
Preconditions: -
Postconditions: Returns name of player(part of set-get methodology)
Invariants: -

getColor

```
public model.characterPack.Characters.charColor getColor()
```

Accessor: Returns color of player (part of set-get methodology)
Preconditions: -
Postconditions: Returns color of player(part of set-get methodology)
Invariants: -

setName

```
public void setName(java.lang.String name)
```

Transformer(Mutative): Sets name of player (part of set-get methodology)
Preconditions: -
Postconditions: Sets name of player (part of set-get methodology)
Invariants: -

setColor

```
public void setColor(model.characterPack.Characters.charColor color)
```

Transformer(Mutative): Sets color of player (part of set-get methodology)
Preconditions: -
Postconditions: Sets color of player (part of set-get methodology)
Invariants: -

PlayerPack

getlastMuseumVisit

```
public int getlastMuseumVisit()
```

Accessor: Returns last museum visit of player (used to determine first move) (part of set-get methodology)
Preconditions: -
Postconditions: Returns last museum visit of player (used to determine first move)(part of set-get methodology)
Invariants: -

getPlayerCharacters

```
public java.util.ArrayList<model.characterPack.Characters> getPlayerCharacters()
```

Accessor: Returns player's characters (part of set-get methodology)
Preconditions: -
Postconditions: Returns player's characters (part of set-get methodology)
Invariants: -

setlastMuseumVisit

```
public void setlastMuseumVisit(int lastMuseumVisit)
```

Transformer(Mutative): Sets last museum visit of player (in terms of days ago) (part of set-get methodology)
Preconditions: -
Postconditions: Sets last museum visit of player (in terms of days ago) (part of set-get methodology)
Invariants: -

addPlayerCharacter

```
public void addPlayerCharacter(Characters character)
```

Transformer(Applicative): Adds a character to the characters that the player owns
Preconditions: -
Postconditions: Adds a character to the characters that the player owns
Invariants: -

getPlayerTiles

```
public java.util.ArrayList<model.tilePack.FindingTile> getPlayerTiles()
```

Accessor: Returns tiles that player owns (part of set-get methodology)
Preconditions: -
Postconditions: Returns tiles that player owns (part of set-get methodology)
Invariants: -

calcPoints

```
public int calcPoints()
```

Accessor: Calculates and returns player's total points
Preconditions: -
Postconditions: Calculates and returns player's total points
Invariants: -

addPlayerTiles

```
public void addPlayerTiles(model.tilePack.FindingTile playerTile)
```

Transformer(Applicative): Adds a tile to the tiles that the player owns
Preconditions: -
Postconditions: Adds a tile to the tiles that the player owns
Invariants: -

getCaryatids

```
public int getCaryatids()
```

Accessor: Returns player's Caryatid tile amount(part of set-get methodology)
Preconditions: When points are to be calculated
Postconditions: Returns player's Caryatid tile amount(part of set-get methodology)
Invariants: -

useCharacter

```
public void useCharacter(model.characterPack.Characters.chars character)
```

Transformer(Mutative): Uses a character that the player owns and is not used already
Preconditions: -
Postconditions: Uses a character that the player owns and is not used already
Invariants: -

getSphinxes

```
public int getSphinxes()
```

Accessor: Returns player's Sphinx tile amount(part of set-get methodology)
Preconditions: When points are to be calculated
Postconditions: Returns player's Sphinx tile amount(part of set-get methodology)
Invariants: -

PlayerPack

useCharacter

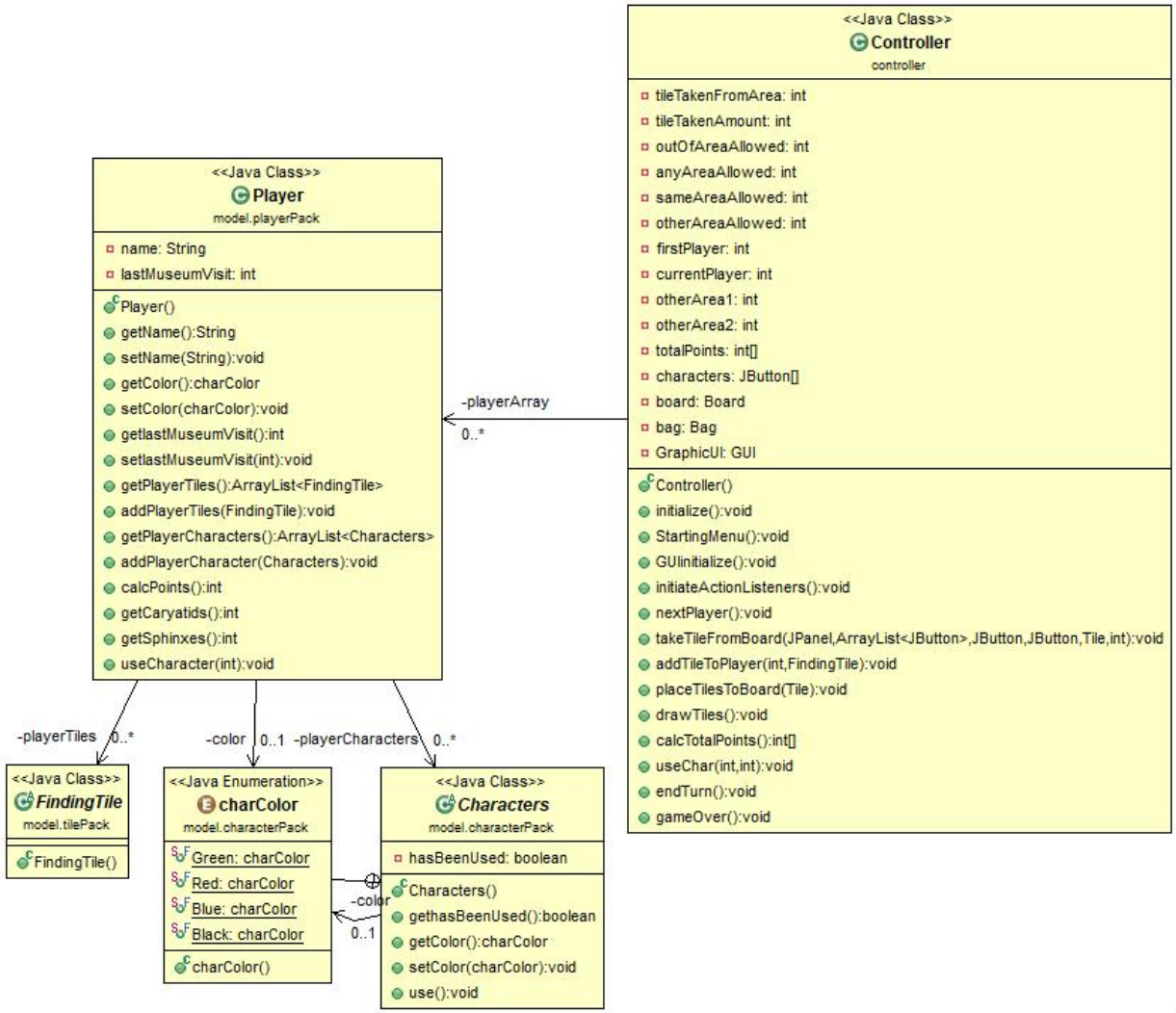
```
public void useCharacter(int character)
```

Transformer(Mutative): Uses a character that the player owns and is not used already
Preconditions: -
Postconditions: Uses a character that the player owns and is not used already
Invariants: -

Phase 2 comments:

I have added a few functions, namely getCaryatids, getSphinxes and useCharacter. getCaryatids and getSphinxes are used for final point calculation in Controller while useCharacter sets the player's character to used.

UML Diagram



TilePack

Tile:

Bag:
-private ArrayList<Tile> bagTiles: is the ArrayList that contains the tiles within the bag (135 spaces)
AmphoraTile:
-private amphoraColor color: holds the color of this specific amphora
-public enum amphoraColor: enumerates the possible colours of the amphora

MosaicTile:
-private mosaicColor color: holds the color of this mosaic tile
-public enum mosaicColor: enumerates the possible colours of mosaic tiles

SkeletonTile:
-private skeletonSize size: holds the size of the skeleton part (big or small)
-private skeletonPart part: holds which part (upper or lower) of the skeleton it is
-public enum skeletonSize: enumerates the possible sizes of the skeleton
-public enum skeletonPart: enumerates the possible parts of the skeleton

Phase 2 comments:

I removed the isAt variables and methods for tiles because they are redundant for my implementation.

TilePack

Bag class

Method Details

fillBag

```
public void fillBag()
```

Transformer(Applicative): Fills the bag with tiles
Preconditions: Game began
Postconditions: Fills the bag with tiles
Invariants: -

takeFromBag

```
public Tile takeFromBag()
```

Transformer(Mutative): Removes a tile from the bag and returns it to caller
Preconditions: Tiles are drawn
Postconditions: Removes a tile from the bag and returns it to caller
Invariants: -

getbagTiles

```
public java.util.ArrayList<Tile> getbagTiles()
```

Accessor: Returns all of the bag's tiles (part of set-get methodology)
Preconditions: -
Postconditions: Returns all of the bag's tiles (part of set-get methodology)
Invariants: -

Tile class

Method Details

No methods in Tile class.

MosaicTile class

Method Details

setmosaicColor

```
public void setmosaicColor(MosaicTile.mosaicColor color)
```

Transformer(Mutative): Sets mosaic's color (part of set-get methodology)
Preconditions: -
Postconditions: Sets mosaic's color (part of set-get methodology)
Invariants: -

getmosaicColor

```
public MosaicTile.mosaicColor getmosaicColor()
```

Accessor: Returns mosaic's color (part of set-get methodology)
Preconditions: -
Postconditions: Returns mosaic's color (part of set-get methodology)
Invariants: -

AmphoraTile class

Method Details

getamphoraColor

```
public AmphoraTile.amphoraColor getamphoraColor()
```

Accessor: Returns amphora's color (part of set-get methodology)
Preconditions: -
Postconditions: Returns amphora's color (part of set-get methodology)
Invariants: -

setamphoraColor

```
public void setamphoraColor(AmphoraTile.amphoraColor color)
```

Transformer(Mutative): Sets amphora's color (part of set-get methodology)
Preconditions:
Postconditions: Sets amphora's color (part of set-get methodology)
Invariants: -

TilePack

SkeletonTile class

Method Details

setSkeletonSize

```
public void setSkeletonSize(SkeletonTile.skeletonSize size)
```

Transformer(Mutative): Sets skeleton's size (part of set-get methodology)
Preconditions: -
Postconditions: Sets skeleton's size (part of set-get methodology)
Invariants: -

setSkeletonPart

```
public void setSkeletonPart(SkeletonTile.skeletonPart part)
```

Transformer(Mutative): Sets skeleton's part (part of set-get methodology)
Preconditions: -
Postconditions: Sets skeleton's part (part of set-get methodology)
Invariants: -

getSkeletonSize

```
public SkeletonTile.skeletonSize getSkeletonSize()
```

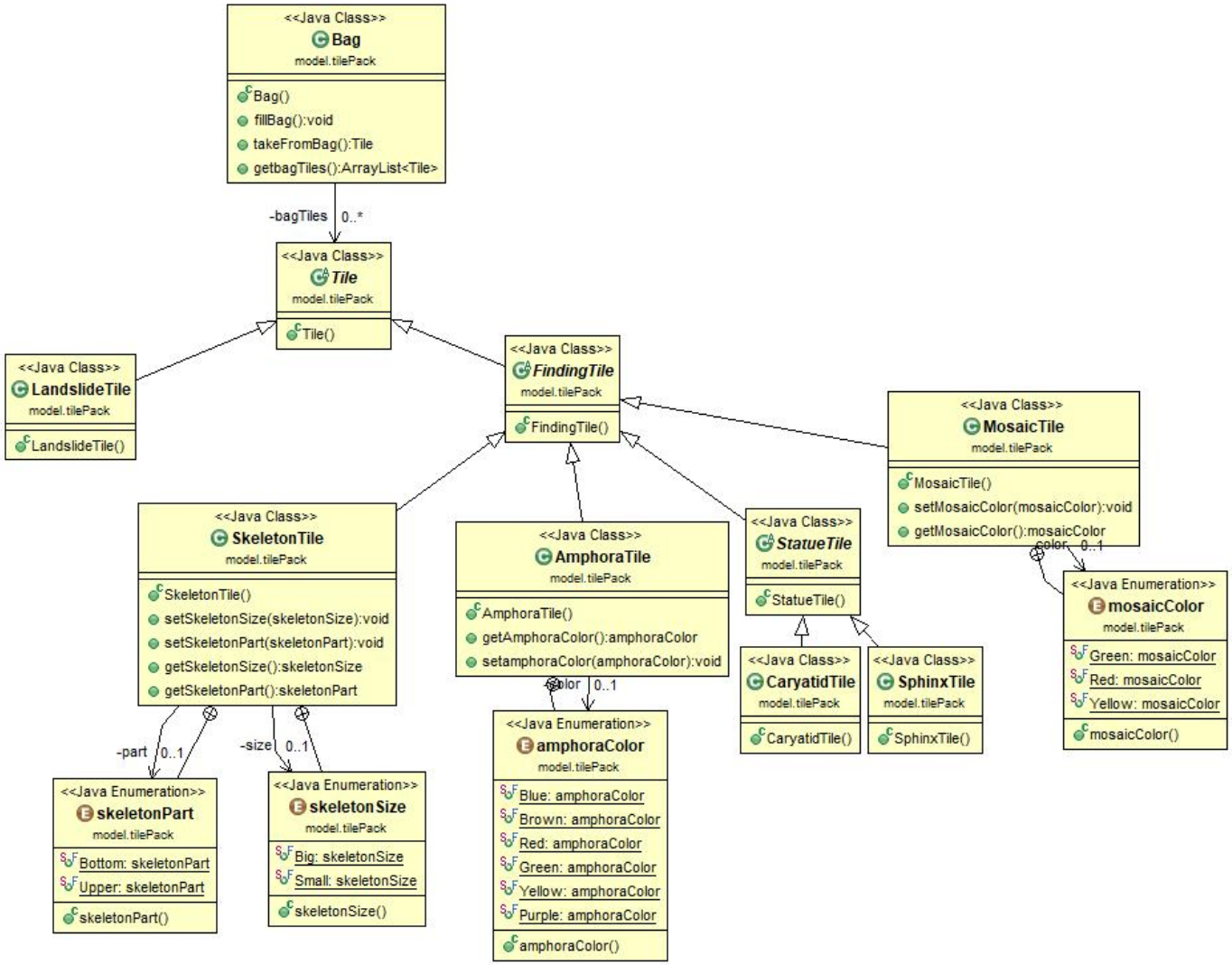
Accessor: Returns skeleton's size (part of set-get methodology)
Preconditions: -
Postconditions: Returns skeleton's size (part of set-get methodology)
Invariants: -

getSkeletonPart

```
public SkeletonTile.skeletonPart getSkeletonPart()
```

TilePack

UML Diagram



CONTROLLER PACK

The Controller is responsible for coordinating the Model with the View components.
Controller

Phase 2 comments:

I have added a LOT of new variables to be used that were needed for my implementation.
Also i have added a lot of classes due to unforeseen circumstances in Phase 1.

```
private int tileTakenFromArea, tileTakenAmount=0, outOfAreaAllowed=0, anyAreaAllowed=0,
sameAreaAllowed=0, otherAreaAllowed=0, firstPlayer, currentPlayer, charHasBeenUsed=0, otherArea1
=-1, otherArea2=-1;
```

All of these other than firstPlayer and currentPlayer are used for calculating availability of taking tiles from board (due to character or the usual two tiles per turn from same area). firstPlayer and currentPlayer are used to cycle through the players.

```
private Player[] playerArray;
```

Used to store the players and their data.

```
private int[] totalPoints;
```

Used for storing points of each player (i needed to make another function other than for calculating final points due to caryatids and sphinx point calculation depending on other players data)

```
private Board board;
private Bag bag;
private GUI GraphicUI;
```

Initializes the classes to be used later

CONTROLLER PACK

Constructor Details

Controller

```
public Controller()
```

Constructor: Runs the initialization functions for the several objects
Preconditions: Game start
Postconditions: Runs the initialization functions for the several objects
Invariants: -

Method Details

initialize

```
public void initialize()
```

Transformer(Mutative): Initializes some variables and sets the game board, players and bag
Preconditions: Game start
Postconditions: Initializes some variables and sets the game board, players and bag
Invariants: -

initiateActionListeners

```
public void initiateActionListeners()
```

Transformer(Mutative): Initializes the action listeners
Preconditions: Game start after GUI initialization
Postconditions: Initializes the action listeners
Invariants: -

GUIinitialize

```
public void GUIinitialize()
```

Transformer(Mutative): Initializes the GUI
Preconditions: Game start after other initializations
Postconditions: Initializes the GUI
Invariants: -

nextPlayer

```
public void nextPlayer()
```

Transformer(Mutative): Gets next player, adjusts the GUI player name field and the player tile field
Preconditions: Every new turn
Postconditions: Gets next player, adjusts the GUI player name field and the player tile field
Invariants: -

StartingMenu

```
public void StartingMenu()
```

Transformer(Mutative): Initializes the Starting Menu
Preconditions: Game start
Postconditions: Initializes the Starting Menu
Invariants: -

CONTROLLER PACK

Method Details

takeTileFromBoard

```
public void takeTileFromBoard(javax.swing.JPanel tileArea,
                              java.util.ArrayList<javax.swing.JButton> tileList,
                              javax.swing.JButton tileType,
                              javax.swing.JButton tileButton,
                              Tile tile,
                              int area)
```

Transformer(Mutative): Removes a tile from the board and adds it to a player's tiles (both GUI and Model components)
Preconditions: A tile has been chosen by the current player (need to have the rights to take it)
Postconditions: Tile is added to player and removed from the board
Invariants: -

addTileToPlayer

```
public void addTileToPlayer(int player,
                            FindingTile tile)
```

Transformer(Mutative) : Adds a specified tile to the current player's tiles (GUI)
Preconditions: When tiles are taken from the board and placed to a player
Postconditions: Adds a specified tile to the current player's tiles (GUI)
Invariants: -

placeTilesToBoard

```
public void placeTilesToBoard(Tile tile)
```

Transformer(Mutative) : Is used to place a tile to the board and add the appropriate action listener to it
Preconditions: every turn when tiles are drawn
Postconditions: Is used to place a tile to the board and add the appropriate action listener to it
Invariants: -

drawTiles

```
public void drawTiles()
```

Transformer(Mutative): Draws tiles from the bag and places them to the board.
Preconditions: every turn
Postconditions: Draws tiles from the bag and places them to the board
Invariants: -

CONTROLLER PACK

Method Details

calcTotalPoints

```
public int[] calcTotalPoints()
```

Transformer(Mutative): Calculates the total points of each player and returns an int array with all the values
Preconditions: end of game
Postconditions: Calculates the total points of each player and returns an int array with all the values
Invariants: -

useChar

```
public void useChar(int player,
                    int character)
```

Transformer(Mutative): Uses a character if he has not already been used and passes the appropriate information
Preconditions: An unused Character has been used after a tile has been drawn from the board and no other character is being used currently
Postconditions: Uses a character if he has not already been used and passes the appropriate information
Invariants: -

endTurn

```
public void endTurn()
```

Transformer(Mutative): Ends the current turn, setting the EndTurn button to disabled and the DrawButton to enabled, clearing all variables related to tiles taken of previous player, and using the nextPlayer function.
Preconditions: Player has clicked the end turn button
Postconditions: Ends the current turn, setting the EndTurn button to disabled and the DrawButton to enabled, clearing all variables related to tiles taken of previous player, and using the nextPlayer function.
Invariants: -

gameOver

```
public void gameOver()
```

Transformer(Mutative): Does the necessary actions upon game ending (calculates points, disposes GUI and initiates the score dialog
Preconditions: 16 Landslide tiles have filled the entry area
Postconditions: Does the necessary actions upon game ending (calculates points, disposes GUI and initiates the score dialog
Invariants: -

View Pack

The GUI class is the main graphic display class of my project. It will be responsible for all interface purposes aside from the Menu and the player number query at the start of the game.

Phase 2 comments:

The GUI has been reworked from the Phase 1 variation, i needed to add quite a few more variables and many methods for it to be modular and to work properly.

private JLayeredPane layeredPane;

The pane on which the background and the tiles are added.

private JPanel backgroundPanel, mosaicArea, statueArea, amphoraArea, skeletonArea, landslideArea, playerTiles, characterPanel, optionsPanel;

The backgroundPanel contains a label background which shows the background image. The Areas are the areas for the tiles to appear. They use a FlowLayout to correctly sort them. The playerTiles panel is the bottom area where the player's tiles appear. It also uses a FlowLayout. Also the Start Game button appears there at the start of the game. The characterPanel contains the player's name, the Use Character label, and the Character buttons. The optionsPanel contains the DrawTiles button and the EndTurn button.

private JButton drawButton, endTurnButton, startGameButton, tileButton;

Variable names are self-explainatory, these are the buttons used.

private ArrayList<JButton> mosaicTiles, statueTiles, amphoraTiles, skeletonTiles, landslideTiles;

ArrayLists that keep a reference of all of the tile buttons.

private JButton[] characters, Archaeologist, Assistant, Digger, Professor;

Button arrays that keep a reference of all the character buttons (does not need to be ArrayList since character amount is static)

View Pack

private ArrayList<JButton>[] allPlayersTiles;

Array of ArrayList (later will be implemented as 4 Array Lists) that keeps an ArrayList for each of the players total tiles.

private JLabel background, useCharacterLabel;

background label contains the background image while useCharacterLabel contains the Use Character text.

private JLabel[] playerNames;

Contains the labels for each of the players' names. (4 labels)

Constructor Details

GUI

```
public GUI()
```

Constructor: Runs the initialization function for the GUI objects
Preconditions: new GUI at game beginning
Postconditions: Runs the initialization function for the GUI objects
Invariants: -

Method Details

initialize

```
public void initialize()
```

Transformer(Applicative): Initializes GUI objects
Preconditions: new GUI at game beginning
Postconditions: Initializes GUI objects
Invariants: -

scale

```
public java.awt.Image scale(java.lang.String URL,
                            int dimension1,
                            int dimension2)
```

Transformer(Mutative): Scales an image to given dimensions
Preconditions: -
Postconditions: Returns scaled image
Invariants: -

View Pack

Method Details

grey

```
public java.awt.Image grey(java.lang.String URL,
                           int dimension1,
                           int dimension2)
```

Transformer(Mutative): Scales an image to given dimensions and greys it out, disabling it
Preconditions: -
Postconditions: Returns scaled and greyed image
Invariants: -

placeTileToArea

```
public javax.swing.JButton placeTileToArea(java.lang.String tileType,
                                           javax.swing.JPanel tileArea,
                                           java.util.ArrayList<javax.swing.JButton> tileList)
```

Transformer(Applicative): Puts specified tile to specified area
Preconditions: -
Postconditions: Puts specified tile to specified area
Invariants: -

removeTileFromArea

```
public void removeTileFromArea(javax.swing.JPanel tileArea,
                               java.util.ArrayList<javax.swing.JButton> tileList,
                               javax.swing.JButton tileType,
                               int tileCount)
```

Transformer(Applicative): Removes specified tile from specified area
Preconditions: -
Postconditions: Removes specified tile from specified area
Invariants: -

addPlayerTile

```
public void addPlayerTile(java.lang.String tileType,
                          int player)
```

Transformer(Applicative): Puts specified tile to specified player's tiles
Preconditions: -
Postconditions: Puts specified tile to specified player's tiles
Invariants: -

View Pack

Method Details

makeNewTileButton

```
public javax.swing.JButton makeNewTileButton(java.lang.String tileType,
                                             int dimensions)
```

Transformer(Applicative): Makes a new tile button with specified icon URL and dimensions
Preconditions: -
Postconditions: Makes a new tile button with specified icon URL and dimensions
Invariants: -

showNextPlayerTiles

```
public void showNextPlayerTiles(int player)
```

Transformer(Applicative)(Mutative): Sets previous player's tiles to not visible and current player's tiles to visible
Preconditions: -
Postconditions: Sets previous player's tiles to not visible and current player's tiles to visible
Invariants: -

initializeCharacter

```
public void initializeCharacter()
```

Transformer(Applicative): Initializes all character buttons
Preconditions: GUI beginning
Postconditions: Initializes all character buttons
Invariants: -

greyCharacter

```
public void greyCharacter(int count)
```

Transformer(Applicative): Greys out specified character
Preconditions: Character has been used
Postconditions: Greys out specified character
Invariants: -

View Pack

Method Details

showNextCharacters

```
public void showNextCharacters(int player)
```

Transformer(Applicative)(Mutative): Sets previous player's characters to not visible and current player's characters to visible
Preconditions: -
Postconditions: Sets previous player's characters to not visible and current player's characters to visible
Invariants: -

setPlayerNames

```
public void setPlayerNames(java.lang.String name1,
                           java.lang.String name2,
                           java.lang.String name3,
                           java.lang.String name4)
```

Transformer(Applicative)(Mutative): Sets the player names
Preconditions: GUI beginning
Postconditions: Sets the player names
Invariants: -

showPlayerName

```
public void showPlayerName(int playercount)
```

Transformer(Mutative): Shows specified player's name
Preconditions: Game beginning and every new turn
Postconditions: Shows specified player's name
Invariants: -

hidePlayerName

```
public void hidePlayerName(int playercount)
```

Transformer(Mutative): Hides specified player's name
Preconditions: Every new turn
Postconditions: Hides specified player's name
Invariants: -

getDrawButton

```
public javax.swing.JButton getDrawButton()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setDrawButton

```
public void setDrawButton(javax.swing.JButton drawButton)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getEndTurnButton

```
public javax.swing.JButton getEndTurnButton()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setEndTurnButton

```
public void setEndTurnButton(javax.swing.JButton endTurnButton)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getCharacterPanel

```
public javax.swing.JPanel getCharacterPanel()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

View Pack

Method Details

getStartGameButton

```
public javax.swing.JButton getStartGameButton()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setStartGameButton

```
public void setStartGameButton(javax.swing.JButton startGameButton)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getPlayerTiles

```
public javax.swing.JPanel getPlayerTiles()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setPlayerTiles

```
public void setPlayerTiles(javax.swing.JPanel playerTiles)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getMosaicArea

```
public javax.swing.JPanel getMosaicArea()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setMosaicArea

```
public void setMosaicArea(javax.swing.JPanel mosaicArea)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getStatueArea

```
public javax.swing.JPanel getStatueArea()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

View Pack

Method Details

getAmphoraArea

```
public javax.swing.JPanel getAmphoraArea()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setAmphoraArea

```
public void setAmphoraArea(javax.swing.JPanel amphoraArea)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getSkeletonArea

```
public javax.swing.JPanel getSkeletonArea()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setSkeletonArea

```
public void setSkeletonArea(javax.swing.JPanel skeletonArea)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getLandslideArea

```
public javax.swing.JPanel getLandslideArea()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setLandslideArea

```
public void setLandslideArea(javax.swing.JPanel landslideArea)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getMosaicTiles

```
public java.util.ArrayList<javax.swing.JButton> getMosaicTiles()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

View Pack

Method Details

getStatueTiles

```
public java.util.ArrayList<javax.swing.JButton> getStatueTiles()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setStatueTiles

```
public void setStatueTiles(java.util.ArrayList<javax.swing.JButton> statueTiles)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getAmphoraTiles

```
public java.util.ArrayList<javax.swing.JButton> getAmphoraTiles()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setAmphoraTiles

```
public void setAmphoraTiles(java.util.ArrayList<javax.swing.JButton> amphoraTiles)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getSkeletonTiles

```
public java.util.ArrayList<javax.swing.JButton> getSkeletonTiles()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setSkeletonTiles

```
public void setSkeletonTiles(java.util.ArrayList<javax.swing.JButton> skeletonTiles)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getLandslideTiles

```
public java.util.ArrayList<javax.swing.JButton> getLandslideTiles()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

View Pack

Method Details

setLandslideTiles

```
public void setLandslideTiles(java.util.ArrayList<javax.swing.JButton> landslideTiles)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getTileButton

```
public javax.swing.JButton getTileButton()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setTileButton

```
public void setTileButton(javax.swing.JButton tileButton)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

getCharacters

```
public javax.swing.JButton[] getCharacters()
```

setCharacters

```
public void setCharacters(javax.swing.JButton[] characters)
```

View Pack

Phase 2 comments:

There will be no "menu" window as stated in Phase 1. Instead, i have added 3 dialog windows, PlayerNamesAndVisits, EndDialog, and ErrorDialog. PlayerNamesAndVisits is launched at start to get input from the player for player names and last time they visited a museum, while the EndDialog is launched when the game ends and shows the players' scores.

PlayerNamesAndVisits

Constructor Details

PlayerNamesAndVisits

```
public PlayerNamesAndVisits()
```

Constructor: Creates a new menu for submitting player names and days since last museum visit
Preconditions: Start of game
Constructor: Creates a new menu for submitting player names and days since last museum visit
Invariants: -

Method Details

submitAction

```
public void submitAction()
```

Transformer(Applicative): Fills the NamesAndVisits String array after checking if values are integers for LastMuseumVisits
Preconditions: Submit button is pressed
Postconditions: Fills the NamesAndVisits String array after checking if values are integers for LastMuseumVisits
Invariants: -

getNamesAndVisits

```
public java.lang.String[] getNamesAndVisits()
```

Accessor: Returns names and visits of players (part of set-get methodology)
Preconditions: After submit has ran
Postconditions: Returns names and visits of players(part of set-get methodology)
Invariants: -

View Pack

Method Details

getSubmit

```
public javax.swing.JButton getSubmit()
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

setSubmit

```
public void setSubmit(javax.swing.JButton submit)
```

Accessor(get), Transformer(Mutative)(set): Part of set/get methodology (same JavaDoc comment for all set-gets)

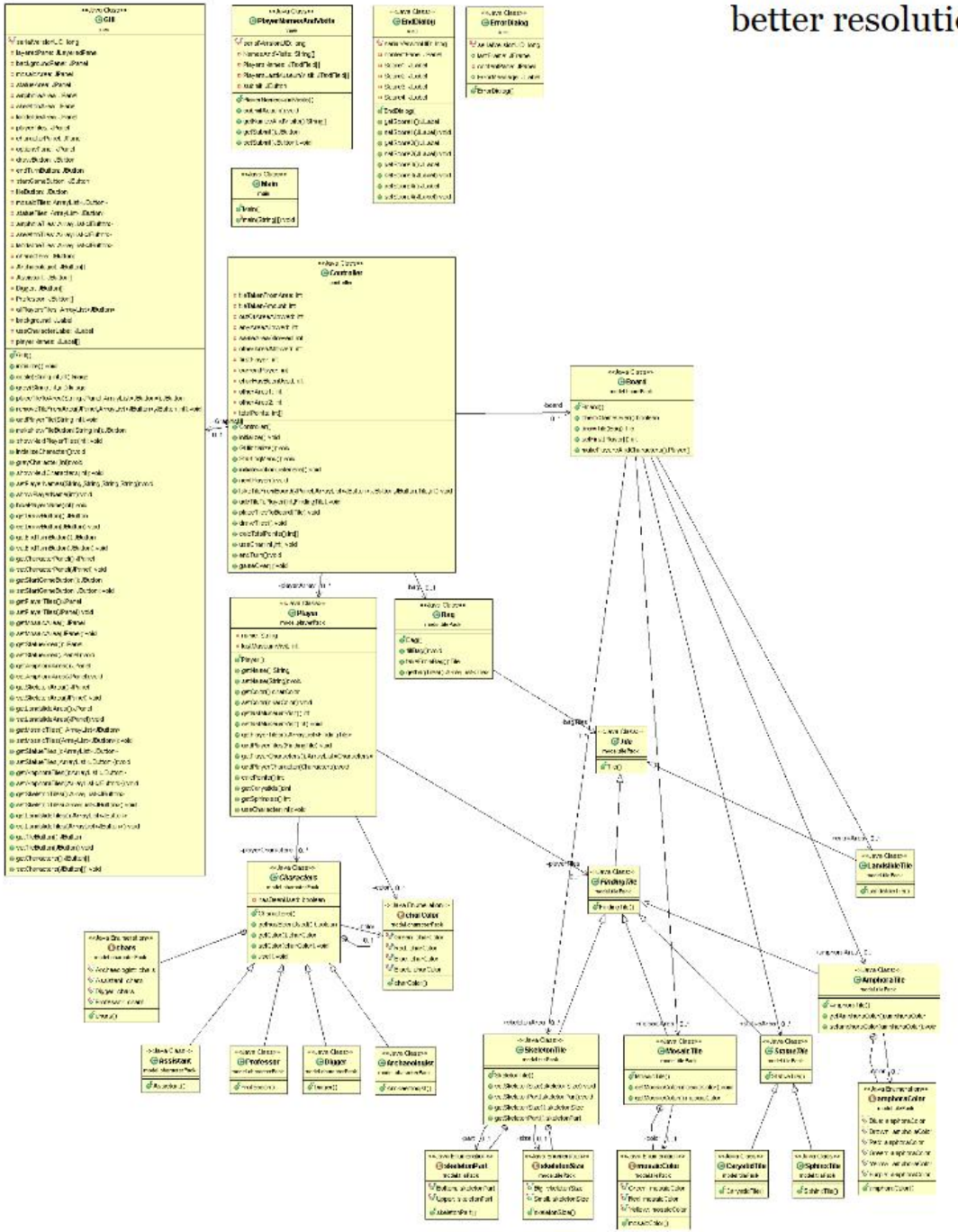
There is also EndDialog and ErrorDialog, basically as i mentioned they are dialog windows that appear at the end of the game, and when an exception happens (player inputs characters instead of int into PlayerNamesAndVisits visit textfields) respectively.

Differences from assignment requirements:

1. I have not implemented saving and the solo player mode.
2. I have made the players' names adjustable and implemented the player turns as per the Amphipolis rulebook; more specifically the player who has visited a museum more recently plays first, and the others rotate after him. This is done through the input that the user gives at the beginning of the game.

Full UML Diagram

(better check inside file for better resolution)



Thanks for reading!