desktop_resources

Class GUI

java.lang.Object
    desktop_resources.GUI

```
public final class GUI
extends java.lang.Object
```

Provides easy access to the GUI features.

Author:

Ronnie Dalsgaard (s093487) with input and adjustments by Daniel Rubin-Grøn (daniel@koru.dk)

Field Summary

Fields

Modifier and Type	Field and Description
`static java.awt.Color`	`BASECOLOR`

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static void`	`addPlayer(java.lang.String name, int balance)` Adds a player to the board. Max.
`static void`	`addPlayer(java.lang.String name, int balance, Car car)` Adds a player to the board. Max.
`static void`	`close()` Closes the GUI, so you can start a new one.
`static void`	`create(Field[] fields)` Initializes the board using an array of fields. Doesn't show the GUI. If this method isn't the first method to be called it will have no result.
`static void`	`displayChanceCard()` Displays the current chance card text in the center.
`static void`	`displayChanceCard(java.lang.String txt)` Sets the text to appear in the center and displays it.
`static java.lang.String`	`getUserButtonPressed(java.lang.String msg, java.lang.String... buttons)` Displays a message to the user and awaits the button pressed response.
`static int`	`getUserInteger(java.lang.String msg)`

	Displays a message to the user and awaits the integer response.
static int	**getUserInteger**(java.lang.String msg, int min, int max) Displays a message to the user and awaits the integer response.
static boolean	**getUserLeftButtonPressed**(java.lang.String msg, java.lang.String trueButton, java.lang.String falseButton) Displays a message to the user and awaits the boolean response.
static java.lang.String	**getUserSelection**(java.lang.String msg, java.lang.String... options) Displays a message to the user and awaits the drop-down response.
static java.lang.String	**getUserString**(java.lang.String msg) Displays a message to the user and awaits the response.
static void	**removeAllCars**(java.lang.String name) Removes all cars belonging to this player.
static void	**removeCar**(int fieldNumber, java.lang.String name) Removes a car from the board. If the car is not on the board, nothing happens.
static void	**removeOwner**(int fieldNumber) Removes an owner from the field. If the field has no owner, nothing happens.
static void	**setBalance**(java.lang.String name, int newBalance) Sets the balance of a player if the player has been added.
static void	**setCar**(int fieldNumber, java.lang.String name) Places a car on the field. All cars can be placed on the same field. A car can only be placed if the corresponding player has been added. If a car is placed on the same field multiple times, nothing more happens. A car can not be placed on multiple fields simultaneously.
static void	**setChanceCard**(java.lang.String txt) Sets the text to appear in the center when calling displayChanceCard() and when the deck is pressed.
static void	**setDescriptionText**(int fieldNumber, java.lang.String description) Sets the Description (The text shown in the center when mouse hovers) of a field on the board.
static void	**setDice**(int faceValue1, int faceValue2) Shows two dice on the board.
static void	**setDice**(int faceValue1, int rotation1, int faceValue2, int rotation2) Shows two dice on the board.
static void	**setDice**(int faceValue1, int x1, int y1, int faceValue2, int x2, int y2) Shows two dice on the board.
static void	**setDice**(int faceValue1, int rotation1, int x1, int y1, int faceValue2, int rotation2, int x2, int y2) Shows two dice on the board.
static void	**setHotel**(int fieldNumber, boolean hasHotel) Sets whether or not a hotel should be on the street and removes all houses if any is present.
static void	**setHouses**(int fieldNumber, int houseCount) Sets houses from the street, and removes the hotel if one is present. If houseCount is out of bounds, nothing happens. If field is not a street, nothing happens.
static void	**setOwner**(int fieldNumber, java.lang.String name) Sets an owner of a field. The field border will have the same color as the player.
static void	**setSubText**(int fieldNumber, java.lang.String subText) Sets the subText of a field on the board.
static void	**setTitleText**(int fieldNumber, java.lang.String title)

	Sets the title of a field on the board.
static void	showMessage(java.lang.String msg) Displays a message to the user. Breaks the system untill "OK" is pressed.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

BASECOLOR

```
public static final java.awt.Color BASECOLOR
```

Method Detail

create

```
public static void create(Field[] fields)
```

Initializes the board using an array of fields.
Doesn't show the GUI.
If this method isn't the first method to be called it will have no result.

Parameters:

`fields` - : Field[]
Fields are created this way:
Field f = new Start.Builder().setTitle("Title").setBgColor(Color.RED).build()
Depending on the type of field, various attributes can be set.
The field types are as follows:
... new Tax.Builder().build();
... new Street.Builder().build();
... new Shipping.Builder().build();
... new Refuge.Builder().build();
... new Jail.Builder().build();
... new Chance.Builder().build();
... new Brewery.Builder().build();
... new Empty.Builder().build();

close

```
public static final void close()
```

Closes the GUI, so you can start a new one.

showMessage

```
public static void showMessage(java.lang.String msg)
```

Displays a message to the user.

Breaks the system untill "OK" is pressed.

Parameters:

`msg` - The message to print.

getUserString

```
public static java.lang.String getUserString(java.lang.String msg)
```

Displays a message to the user and awaits the response.

Parameters:

`msg` - The message that promts the user.

Returns:

The string that the user has entered.

getUserInteger

```
public static int getUserInteger(java.lang.String msg,
                                 int min,
                                 int max)
```

Displays a message to the user and awaits the integer response. Only values between min and max are allowed.

Parameters:

`msg` - The message that promts the user.

`min` - The minimum value the user is allowed to enter.

`max` - The maximum value the user is allowed to enter.

Returns:

The integer that the user selected.

getUserInteger

```
public static int getUserInteger(java.lang.String msg)
```

Displays a message to the user and awaits the integer response.

Parameters:

`msg` - The message that promts the user.

Returns:

The integer that the user selected.

getUserButtonPressed

```
public static java.lang.String getUserButtonPressed(java.lang.String msg,
                                                    java.lang.String... buttons)
```

Displays a message to the user and awaits the button pressed response.

Parameters:

`msg` - The message that promts the user.

`buttons` - A number of strings that should be printed on the buttons the user can press.

Returns:

The string from the button that the user pressed.

getUserSelection

```
public static java.lang.String getUserSelection(java.lang.String msg,
                                                 java.lang.String... options)
```

Displays a message to the user and awaits the drop-down response.

Parameters:

`msg` - The message that promts the user.

`options` - A number of strings with the texts that the user should choose from.

Returns:

The string that the user selected.

getUserLeftButtonPressed

```
public static boolean getUserLeftButtonPressed(java.lang.String msg,
                                               java.lang.String trueButton,
                                               java.lang.String falseButton)
```

Displays a message to the user and awaits the boolean response.

Parameters:

`msg` - The message that promts the user.

`trueButton` - The text that should appear on the left button.

`falseButton` - The text that should appear on the right button.

Returns:

True if the left button is pressed by the user. False otherwise.

setTitleText

```
public static void setTitleText(int fieldNumber,
                                java.lang.String title)
```

Sets the title of a field on the board.

Parameters:

`fieldNumber` - : int [1:40]

`title` - : String (Mind the length!)

setSubText

```
public static void setSubText(int fieldNumber,
```

```
                                java.lang.String subText)
```

Sets the subText of a field on the board.

Parameters:

`fieldNumber` - : int [1:40]

`subText` - : String (Mind the length!)

setDescriptionText

```
public static void setDescriptionText(int fieldNumber,
                                      java.lang.String description)
```

Sets the Description (The text shown in the center when mouse hovers) of a field on the board.

Parameters:

`fieldNumber` - : int [1:40]

`description` - : String (Mind the length!)

addPlayer

```
public static void addPlayer(java.lang.String name,
                             int balance,
                             Car car)
```

Adds a player to the board.
Max. 6 players.

Parameters:

`name` - : String (Mind the length!) (Unique identifier of the player - no duplicates)

`balance` - : int

`car` - : Car Cars are created this way:
Car car = new Car.Builder()
.primaryColor(Color.MAGENTA)
.secondaryColor(Color.BLUE)
.typeTractor()
.patternDotted()
.Build();

addPlayer

```
public static void addPlayer(java.lang.String name,
                             int balance)
```

Adds a player to the board.
Max. 6 players.

Parameters:

`name` - : String (Mind the length!) (Unique identifier of the player - no duplicates)

`balance` - : int

setBalance

```
public static void setBalance(java.lang.String name,
                              int newBalance)
```

Sets the balance of a player if the player has been added.

Parameters:

`name` - The name of the player

`newBalance` - : int

setDice

```
public static void setDice(int faceValue1,
                           int rotation1,
                           int x1,
                           int y1,
                           int faceValue2,
                           int rotation2,
                           int x2,
                           int y2)
```

Shows two dice on the board. The dice will have the specified values, but the placement is random.

Parameters:

`faceValue1` - : int [1:6]

`rotation1` - : int [0:360]

`faceValue2` - : int [1:6]

`rotation2` - : int [0:360[

`x1` - : int [0:11]

`y1` - : int [0:11]

`x2` - : int [0:11]

`y2` - : int [0:11]
(If a parameter is out of bounds nothing will happen!)

setDice

```
public static void setDice(int faceValue1,
                           int x1,
                           int y1,
                           int faceValue2,
                           int x2,
                           int y2)
```

Shows two dice on the board. The dice will have the specified values, but the placement is random.

Parameters:

`faceValue1` - : int [1:6]

`faceValue2` - : int [1:6]

`x1` - : int [0:11]

`y1` - : int [0:11]

`x2` - : int [0:11]

`y2` - : int [0:11]
(If a parameter is out of bounds nothing will happen!)

setDice

```
public static void setDice(int faceValue1,
                           int rotation1,
                           int faceValue2,
                           int rotation2)
```

Shows two dice on the board. The dice will have the specified values, but the placement is random.

Parameters:

`faceValue1` - : int [1:6]

`rotation1` - : int [0:360]

`faceValue2` - : int [1:6]

`rotation2` - : int [0:360[
(If a parameter is out of bounds nothing will happen!)

setDice

```
public static void setDice(int faceValue1,
                           int faceValue2)
```

Shows two dice on the board. The dice will have the specified values, but the placement is random.

Parameters:

`faceValue1` - : int [1:6]

`faceValue2` - : int [1:6]
(If a parameter is out of bounds nothing will happen!) Uses random rotation.

displayChanceCard

```
public static void displayChanceCard(java.lang.String txt)
```

Sets the text to appear in the center and displays it.

Parameters:

`txt` - : String (Mind the length!)

setChanceCard

```
public static void setChanceCard(java.lang.String txt)
```

Sets the text to appear in the center when calling displayChanceCard() and when the deck is pressed.

Parameters:

`txt` - : String (Mind the length!)

displayChanceCard

```
public static void displayChanceCard()
```

Displays the current chance card text in the center.

setCar

```
public static void setCar(int fieldNumber,
                          java.lang.String name)
```

Places a car on the field.
All cars can be placed on the same field.
A car can only be placed if the corresponding player has been added.
If a car is placed on the same field multiple times, nothing more happens.
A car can not be placed on multiple fields simultaneously.

Parameters:

`fieldNumber` - : int [1:40]

`name` - The name of the player

removeCar

```
public static void removeCar(int fieldNumber,
                             java.lang.String name)
```

Removes a car from the board.
If the car is not on the board, nothing happens.

Parameters:

`fieldNumber` - : int [1:40]

`name` - The name of the player

removeAllCars

```
public static void removeAllCars(java.lang.String name)
```

Removes all cars belonging to this player.

Parameters:

`name` - The name of the player.

setOwner

```
public static void setOwner(int fieldNumber,
                            java.lang.String name)
```

Sets an owner of a field.
The field border will have the same color as the player. The owners name will be printed in the subText. If the field is not a street, shipping or a brewery nothing happens.
If the owner hasn't been added to the board, nothing happens.

Parameters:

`fieldNumber` - : int [1:40]

`name` - The name of the player

removeOwner

```
public static void removeOwner(int fieldNumber)
```

Removes an owner from the field.
If the field has no owner, nothing happens.

Parameters:

`fieldNumber` - : int [1:40]

setHouses

```
public static void setHouses(int fieldNumber,
                             int houseCount)
```

Sets houses from the street, and removes the hotel if one is present.
If houseCount is out of bounds, nothing happens.
If field is not a street, nothing happens.

Parameters:

`fieldNumber` - : int [1:40]

`houseCount` - : int [0:4]

setHotel

```
public static void setHotel(int fieldNumber,
                            boolean hasHotel)
```

Sets whether or not a hotel should be on the street and removes all houses if any is present.

Parameters:

`fieldNumber` - : int [1:40]

`hasHotel` - : boolean