Class	VariableName : type	Comment	
AddBooks	availble : boolean	//Describe if the book is available or not	
	book : Books	//Represent a book	
	centerPanel : JPanel	//Represent the panel	
	data : String[]	//Represent the data of the book addiction	
	informationLabel : JLabel[]	//Represent the label of information	
	informationLabelPanel : JPanel	//Represent the information label panel	
	informationString : String[]	//Represent the information informed by the user	
	informationTextField : JTextField[]	//Represent the information textField	
	informationTextFieldPanel : JPanel	//Represent the text field panelto write the information	
	insertInformationButton : JButton	//Represent the button to insert the information	
	insertInformationButtonPanel : JPanel	//Represent the panel to the insert information button	
	lblShelfNo : JLabel		
	northLabel : JLabel		
	northPanel : JPanel		
	OKButton : JButton	// Represent de OK button	
	southPanel : JPanel		
	txtShelfNo : JTextField		
AddMembers	centerPanel : JPanel	//Represent the center panel	
	data : String[]	//Represent the data of the addiction operation	
	informaionString : String[]	//Represent the information informed by the user	
	informationLabel : JLabel[]	//Represent the label where the user isert the information	
	informationLabelPanel : JPanel	//Represent the panel of the label information	
	informationPasswordField : JPasswordField[]	//Represent the information Password field	
	informationTextField : JTextField[]	//Represent the information text field where the user put the information	
	informationTextFieldPanel : JPanel	//Represent the information text field panel	
	insertInformationButton : JButton	//Represent insert information button	
	insertInformationButtonPanel : JPanel	//Represent the insert information button panel	
	member : Members	//Represent the member	
	northLabel : JLabel	//Create the North Label	
	northPanel : JPanel	//Create the North Panel	
	OKButton : JButton	//Create the OK button	
	southPanel : JPanel	//Create the South Panel	
Books	author : String	//Represent the author of the book	
	availble : boolean	//Inform if the book is available or not	
	bookID : int	//Represent the book ID	
	connection : Connection	//Represent the connection status	

	copyright : int	//Represent the copyright	
	edition : int	//Inform the edition of the book	
	ISBN : String		
	library : String	//Inform the name of the library	
	numberOfAvailbleBooks : int	//Inform the number of books available	
	numberOfBooks : int	//Inform the number of books in the library	
	numberOfBorrowedBooks : int	//Inform the number of borrowed books	
	pages : int	//Inform the number of pages of the book	
	publisher : String	//Represent the name of the publisher	
	resultSet : ResultSet		
	statement : Statement		
	subject : String		
	title : String	//Represent the title of the book	
	URL : String		
Borrow	bookID : int	//Inform the book ID	
	connection : Connection	//Inform the connection status	
	dayOfBorrowed : Date	//Represent the day of the borrowing	
	dayOfReturn : Date	//Represent the day of the return	
	memberID : int	//Represent the member ID	
	resultSet : ResultSet		
	statement : Statement		
	URL : String		
BorrowBooks	book : Books	//Inform the book that is going to be borrowed	
	borrow : Borrow	//Represent the borrowing	
	borrowButton : JButton	//Represent the borrow button	
	borrowButtonPanel : JPanel	//Represent the borrow button panel	
	cancelButton : JButton	//Represent the cancel button	
	centerPanel : JPanel		
	data : String[]		
	date : String		
	informationLabel : JLabel[]		
	informationPanel : JPanel		
	informationString : String[]		
	informationTextField : JTextField[]		
	member : Members	//Represent the member	
	northPanel : JPanel		
	southPanel : JPanel		
	title : JLabel	//Inform the title of the book that is going to be borrowed	

Center	l : JLibrary		
JLibrary	addBooks : AddBooks	//creates objects from others classes to use in the ActionListener	
	addMembers : AddMembers	//creates objects from others classes to use in the ActionListener	
	borrowBooks : BorrowBooks	//creates objects from others classes to use in the ActionListener	
	desktop : JDesktopPane	//variable to add objects from others classes	
	desktopScrollPane : JScrollPane	//does nothing...	
	goButton : JButton	//represents a button in the search tool bar	
	listBooks : ListBooks	//creates objects from others classes to use in the ActionListener	
	listMembers : ListMembers	//creates objects from others classes to use in the ActionListener	
	menu : Menubar	//creates objects from others classes to use in the ActionListener	
	returnBooks : ReturnBooks	//creates objects from others classes to use in the ActionListener	
	search : SearchBooksAndMembers	//creates objects from others classes to use in the ActionListener	
	searchLabel : JLabel	//represents a label in the search tool bar	
	searchPanel : JPanel	//represents a panel in the search tool bar	
	searchTextField : JTextField	//represents a text field in the search tool bar	
	searchToolBar : JToolBar	//represents a toolbar in the search tool bar	
	splitPane : JSplitPane	//does nothing...	
	statusbar : StatusBar	//creates objects from others classes to use in the ActionListener	
	toolbar : Toolbar	//creates objects from others classes to use in the ActionListener	
	treeScrollPane : JScrollPane	//does nothing...	
ListBooks	DATABASE_URL : String	//constant of the database url	
	DEFAULT_QUERY : String		
	JDBC_DRIVER : String	//database related	
	centerPanel : JPanel	//for the creation of center panel	
	column : TableColumn	//for the creation of the tablem column	
	northLabel : JLabel	//label creation	
	northPanel : JPanel	//panel creation	
	printButton : JButton	//button creation	
	scrollPane : JScrollPane	//scroll pane creation	
	table : JTable	//table creation	
	tableModel : ResultSetTableModel	//storing an object of the ResultSetTableModel class	
ListMembers	DATABASE_URL : String	//constant of the database url	
	DEFAULT_QUERY : String		
	JDBC_DRIVER : String	//database related	
	centerPanel : JPanel	//for the creation of center panel	
	column : TableColumn	//for the creation of the tablem column	

	label : JLabel	//label creation	
	northPanel : JPanel	//panel creation	
	printButton : JButton	//button creation	
	scrollPane : JScrollPane	//scroll pane creation	
	table : JTable	//table creation	
	tableModel : ResultSetTableModel	//storing an object of the ResultSetTableModel class	
ListSearchBooks	centerPanel : JPanel	//panel creation	
	column : TableColumn	//Table Column creation	
	label : JLabel	//label creation	
	northPanel : JPanel	//panel creation	
	printButton : JButton	//button creation to print in the search	
	scrollPane : JScrollPane	//scroll pane creation	
	table : JTable	//table creation	
	tableModel : ResultSetTableModel	//storing an object of the ResultSetTableModel class	
ListSearchMembers	centerPanel : JPanel	//panel creation	
	column : TableColumn	//Table Column creation	
	label : JLabel	//label creation	
	northPanel : JPanel	//panel creation	
	printButton : JButton	//button creation to print in the search	
	scrollPane : JScrollPane	//scroll pane creation	
	table : JTable	//table creation	
	tableModel : ResultSetTableModel	//storing an object of the ResultSetTableModel class	
Members	connection : Connection	//Connection status	
	email : String	//Member's email	
	expired : Date	//Date of the expiration	
	ID : int	//Member ID	
	major : String		
	memberID : int		
	mony : int		
	name : String	//Member's name	
	numberOfBooks : int	//Number of books borrowed by this member	
	password : String	//Member's password	
	resultSet : ResultSet		
	statement : Statement		
	URL : String		
Menubar	addBook : JMenuItem	//Book addition	

	addMember : JMenuItem	//Member addition	
	bookMenu : JMenu	//Represent the book menu	
	borrowBook : JMenuItem	//Book borrowing	
	exit : JMenuItem	//Exiting the menu	
	fileMenu : JMenu	//Menu creation	
	icons : ImageIcon[]	//Icon creation	
	imageName16 : String[]	//	
	listBook : JMenuItem	//List the books	
	listMember : JMenuItem	//List the members	
	loanMenu : JMenu	//Create the loan menu	
	memberMenu : JMenu	//Create member menu	
	returnBook : JMenuItem	//Returning a book	
	searchBooksAndMembers : JMenuItem	//Searching books and members	
	searchMenu : JMenu	//Create the seach menu	
PrintingBooks	TAB_SIZE : int	//Tab size	
	connection : Connection	//Connection status	
	lines : Vector	//Create the lines	
	resultset : ResultSet	//Resultset from the statement which comes from the data base	
	statement : Statement	//Creating the statement	
	textArea : JTextArea	//Create the text area	
	URL : String	//Constant of the URL from the database	
PrintingMembers	TAB_SIZE : int	//Constant to check number os spaces in the Vector method	
	connection : Connection	//Connection status	
	lines : Vector	//Create the lines	
	resultset : ResultSet	//Resultset from the statement which comes from the data base	
	statement : Statement	//Creating the statement	
	textArea : JTextArea	//Creating text area	
	URL : String	//Constant of the URL from the database	
ResultSetTableModel	connectedToDatabase : boolean	//Keep track of database connection status	
	connection : Connection	//Connection status	
	metaData : ResultSetMetaData	//Getting the metaData from the ResultSet Class	
	numberOfRows : int	//Number of rows from the ResultSet	
	resultSet : ResultSet	//Resultset from the statement which comes from the data base	
	statement : Statement	//Creating the statement	
ReturnBooks	book : Books	//Represent a book	
	borrow : Borrow	//Represent the borrowing	

	cancelButton : JButton	//Create the cancel button	
	centerPanel : JPanel	//Create the center panel	
	data : String[]	//Data of the returning	
	informationLabel : JLabel[]	//Create the information label	
	informationPanel : JPanel	//Create the information panel	
	informationString : String[]	//Information string	
	informationTextField : JTextField[]	//Creating information text field	
	lblFinePerDay : JLabel		
	lblTotalFineAmt : JLabel		
	member : Members	//Member that is returning a book	
	northPanel : JPanel	//Create a north panel	
	returnButton : JButton	//Create return button	
	returnButtonPanel : JPanel	//Create return button panel	
	southPanel : JPanel	//Create south panel	
	title : JLabel	//Book title	
	txtFinePerDay : JTextField		
	txtTotalFineAmt : JTextField		
SearchBooksAndMembers	book : Books	//Book to be searched	
	booksData : String[]	//Create an array of sting to store data	
	booksKey : JLabel	//Create a label for the book key	
	booksKeyTextField : JTextField	//Create the book key textfield	
	booksTypes : String[]		
	cancelButton : JButton	//Create the cancel button	
	center : JPanel	//Create center	
	centerBooksPanel : JPanel	//Create center panel	
	centerMembersPanel : JPanel	//Create the Center Panel	
	listBooks : ListSearchBooks	//Create a list of books	
	listMembers : ListSearchMembers	//Create a list of members	
	member : Members	//Create a member	
	membersData : String[]	//Create an array to store data	
	membersKey : JLabel	//Create the label for the key	
	membersKeyTextField : JTextField	//Create the text field	
	membersTypes : String[]	//Create an array of strings	
	northPanel : JPanel	//Create north panel	
	searchBooksButton : JButton	//Create the button search	
	searchBooksButtonPanel : JPanel	//Create an Internal Panel in the center panel	
	searchBooksLabel : JLabel	//Create the search book label	
	searchBooksPanel : JPanel	//Create the search book panel	
	searchBooksTypes : JComboBox	//Create the serach book combo box	

	searchMembersButton : JButton	//Create the members button	
	searchMembersButtonPanel : JPanel	//Create an Internal Panel in the center panel	
	searchMembersLabel : JLabel	//Create the table	
	searchMembersPanel : JPanel	//Create an Internal Panel in the center panel	
	searchMembersTypes : JComboBox		
	southPanel : JPanel	//Create the south panel	
	title : JLabel	//For criating the tile label	
StatusBar	statusBar : JLabel	//Create a label	
Toolbar	button : JButton[]	//Create the buttons to use them in ToolBar	
	imageName24 : String[]	//Create the name of the image file 24*24	
	tipText : String[]	//Create the tipText for the toolbar	
xxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
Entity	image : BufferedImage	//Image in buffered	
	loc : Node	//Represent on place in the map	
	map : Map	//Represent the map	
	seenByPlayer : boolean	//Describe if the player can see or not	
	visibleToPlayer : boolean	//Describe if can be visible for the player or not	
Actor	acted : boolean	//Represent if something was done in the player's turn	
	atk : int	//Represent the attack force	
	attacked : boolean	//Describe if the player was attacked or not	
	damageDealt : int	//Quantity of damage caused in other	
	damageTaken : int	//Quantity of damage caused in player	
	hp : int	//Life of the Player	
	initialPathSize : int	//Actual place of the player in map	
	level : int	//Represent the Player's level	
	maxHP : int	//Represent maximum amount of hp	
	moved : boolean	//Describe if the player was moved or not	
	myTurn : boolean	//Describe if the current turn is the Player's turn	
	path : List<Node>	//List contains all map's places	
	previousNode : Node	//Saves the previous place	
	sm : SoundManager	//Sound Manager	
	stance : boolean[]	//Describes the conditions: normal, attacking, hit and shooting	
	timers : int[]	//Represent the timers of the game	
	turnsOnNode : int	//Turns on node	

ActorQueue	queue : List<Actor>	//Represent the queue of the Actors	
Enemy	awareOfPlayer : boolean	//Describe if the enemy is awere of player	
	targeted : boolean	//Describe if the enemey is he target	
	xpReward : int	//Represent the amount of experience reward	
EnemyGroup	enemies : List<Enemy>	//List of enemys in the map	
	map : Map	//Represents the game's map	
Player	blink : boolean	//Represents blink state of the player	
	cancel : boolean	//Represents cancel state (action on target) of the player	
	close : boolean	//Represents the action to close	
	dl : boolean	//Represents the "dl" direction	
	dn : boolean	//Represents the "dn" direction	
	dr : boolean	//Represents the "dr" direction	
	enemiesDefeated : int	//Counts the total of enemies defeated	
	explode : boolean	//Action from the wizzard character	
	fireArrow : boolean	//Action of the the ranger character	
	getTargets : boolean	//Action to interect with target	
	gotTargets : boolean	//Target got	
	hasKey : boolean	//Checks if the player has the key	
	lt : boolean	//Represents the "lt" direction	
	newEnemies : boolean	//Represents the enemies visible	
	nextTarget : boolean	//Represents the order of the actions on the targets	
	openedLock : boolean	//Checks if a lock is opened	
	previousTarget : boolean	//Represents the order of the actions on the targets	
	quicken : boolean	//Action from the wizzard character	
	rt : boolean	//Represents the "rt" direction	
	shout : boolean	//Shout action of the player	
	target : Enemy	//Represents the enemy which the player can interect	
	targetEnemy : boolean		
	targets : List<Enemy>	//Represents the list of enemies	
	turnsOnLevel : int	//Represents the regen of the health	
	turnsSinceCombat : int	//Represents the regen on battle of the barbarian character	
	ul : boolean	//Represents the ul direction	
	up : boolean	//Represents the "up" direction	
	ur : boolean	//Represents the "ur" direction	
	wait : boolean	//Represents the ending of turn witout action	
	xpEarned : double	//Represents the total amount of xp from the player	

Ranger	targets : List<Enemy>		
GameplayState	addEnemyInterval : int	//Number of turns after which an enemy is added	
	aq : ActorQueue		
	dungeonLevel : int	//Level of the dungeon	
	eg : EnemyGroup	//Group of enemies in the map	
	map : Map	//Represents the map	
	player : Player	//Represents the player	
	tickEnemies : boolean		
	tickTimer : int		
	TURN_DELAY : int		
	turnCounter : int	//Number of turns	
	classChoice : int	//Represents the chosen class	
GameStateManager	GAMEPLAY_STATE : int	//Number that represent the gameplay state	
	MENU_STATE : int	//Number that represent the menu state	
	current : GameState	//The current game state	
	gameStates : List<GameState>	//List of game states	
LoseGameState	down : boolean	//Command used to change the choice	
	enter : boolean	//Command used to confirm the choice	
	up : boolean	//Command used to change the choice	
	choice : int	//Number that represents the coices	
	choices : String[]	//Choices in the end of the game (yes or no)	
	enemiesDefeated : int	//Number of enemies defeated in the game	
MenuState	down : boolean	//Change the charactr's class	
	enter : boolean	//Confirm the charactr's class	
	FONT_SIZE : int	//Size of the font	
	up : boolean	//Change the character's class	
	classChoice : int	//Class choosen by the user	
	menuOptions : String[]	//Options of character's class (barbarian, thief, etc.)	
WinGameState	down : boolean	//Command used to change the choice	
	enter : boolean	//Command used to confirm the choice	
	up : boolean	//Command used to change the choice	
	choice : int	//Number that represents the coices	
	choices : String[]	//Choices in the end of the game (yes or no)	
	enemiesDefeated : int	//Number of enemies defeated in the game	

	turns : int	//Number of turns	
ImageManager	ant : BufferedImage	//Ant image	
	archer : BufferedImage	//Archer image	
	arrows : BufferedImage	//Arrows image	
	bang : BufferedImage	//Bang image	
	barbarian : BufferedImage	//Barbarian image	
	bat : BufferedImage	//Bat image	
	boundaryWall : BufferedImage	//Image of boundary Wall	
	closedDoor : BufferedImage	//Image of coled door	
	closedDoorShadow : BufferedImage	//Shadow image of the closed door	
	corpse : BufferedImage	//Corpse image	
	demon : BufferedImage	//Demon image	
	eye : BufferedImage	//Eye image	
	fire : BufferedImage	//Fire image	
	frogKnight : BufferedImage	//Image of the Frog Knight	
	gargoyle : BufferedImage	//Gargoyle image	
	gelatinousCube : BufferedImage	//Gelatinous Cube image	
	goal : BufferedImage	//Goal image	
	goalShadow : BufferedImage	//Shadow image of the goal	
	goblin : BufferedImage	//Goblin image	
	gremlin : BufferedImage	//Gremilin image	
	hammer : BufferedImage	//Hammer image	
	hammerShadow : BufferedImage	//Shadow image of the hammer	
	imp : BufferedImage	//Imp image	
	key : BufferedImage	//Key image	
	keyShadow : BufferedImage	//Shadow image of the key	
	lock : BufferedImage	//Lock image	
	lockOpen : BufferedImage	//Image of open lock	
	lockOpenShadow : BufferedImage	//Shadow image of the open lock	
	lockShadow : BufferedImage	//Shadow image of the lock	
	miss : BufferedImage	//Miss image	
	ogre : BufferedImage	//Ogre image	
	openDoor : BufferedImage	//Image of the open door	
	openDoorShadow : BufferedImage	//Shadow image of the open door	
	panther : BufferedImage	//Panther image	
	potion : BufferedImage	//Potion image	
	potionShadow : BufferedImage	//Shadow image of the potion	
	ranger : BufferedImage	//Ranger image	
	rat : BufferedImage	//Rat image	

	scorpion : BufferedImage	//Scorpion image	
	snake : BufferedImage	//Snake image	
	spider : BufferedImage	//Spider image	
	stairDown : BufferedImage	//Image of stair down	
	stairDownShadow : BufferedImage	//Shadow image of the stair down	
	stoneTile : BufferedImage	//Stone Tile image	
	stoneTile1Shadow : BufferedImage	//Another stile of stone tile image	
	stoneWall : BufferedImage	//Stone wall image	
	stoneWallShadow : BufferedImage	//Shadow image of the stone wall	
	swords : BufferedImage	//Swords image	
	thief : BufferedImage	//Thief imade	
	voidNode : BufferedImage	//Empty place image	
	wasp : BufferedImage	//Wasp image	
	wizard : BufferedImage	//Wizard image	
	wolf : BufferedImage	//Wolf image	
	worm : BufferedImage	//Worm image	
	wyvern : BufferedImage	//Wyvern image	
	zombie : BufferedImage	//Zombie image	
SpriteSheet	sheet : BufferedImage	//Sheet image	
Game	COLUMNS : int	//Columns on the map	
	fps : double		
	gameThread : Thread		
	gsm : GameStateManager	//Menage the game states	
	im : ImageManager	//Menage the images on the screen	
	MAX_FRAME_SKIPS : int	//The maximum of frames that can bo skiped	
	maxUpdateTime : double	//Represents the maximum time to update	
	MSG_HEIGHT : int		
	NO_DELAYS_PER_YIELD : int		
	ROWS : int	//Row on the map	
	running : boolean	//Repesents if the game is running or not	
	SCALE : int		
	SCALED_TILE_SIZE : int		
	serialVersionUID : long		
	TARGET_FPS : int		
	tickTimer : int		
	TILE_SIZE : int	//Represent the size of a tile	
	TURN_DELAY : int		
	turnCounter : int	//Number of turns	

	W_COLS : int	//Displayed nodes	
	W_HEIGHT : int	//Window height	
	W_ROWS : int	//Displeyed nodes	
	W_WIDTH : int	//Window width	
	spriteSheet : BufferedImage		
	ticksPerSecond : double		
KeyManager	gsm : GameStateManager		
Map	displayedNodesMaxX : int	//Maximum of nodes in the x axis	
	displayedNodesMaxY : int	//Maximum of nodes in the y axis	
	displayedNodesMinX : int	//Minimum of nodes in the x axis	
	displayedNodesMinY : int	//Minimum of nodes in the y axis	
	MAX_ROOM_HEIGHT : int	//Maximum height of a room	
	MAX_ROOM_WIDTH : int	//Maximum width of a room	
	MIN_ROOM_HEIGHT : int	//Minimum height of a room	
	MIN_ROOM_WIDTH : int	//Minimum width of a room	
	displayedNodes : Node[][]	//Nodes to be displayed	
	endNodeFound : boolean	//Value representing the last node	
	hSize : int	//Columns in the game	
	imageMap : BufferedImage[][]		
	mapGrid : Node[][]		
	rooms : Room[][]	//Rooms in the map	
	visibleToPlayer : List<Node>	//Nodes that the pleyer can see	
	vSize : int	//Rows in the game	
Node	closedDoor : Feature	//Representation of the closed door space	
	floor : Feature	//Representation of the floor space	
	openDoor : Feature	//Representation of the open door space	
	voidNode : Feature	//Representation of the empty place	
	wall : Feature	//Representation of the wall place	
	entities : List<Entity>	//List of entities space	
	feature : Feature	//Representation of any kind of feature (places)	
	fScore : int	//Representation of the moviment	
	gScore : int	//Representation of the moviment	
	hScore : int	//Representation of the moviment	
	image : BufferedImage	//Image of the node	
	isDoorClosed : boolean	//Define if the Door node is closed	
	isDoorOpen : boolean	//Define if the Door node is open	
	isFloor : boolean	//Define if the node is floor	

	isVoid : boolean	//Define if the node is empty	
	isWall : boolean	//Define if the node is wall	
	map : Map	//Representation of the game's map	
	parentNode : Node	//Representetion of the parent node	
	seenByPlayer : boolean	//Describe if the player can see or not the node	
	x : int	//Coordinate x	
	y : int	//Coordinate y	
Room	boundary : Rectangle	//Representation of the boundarie of the map	
	column : int	//Represents the column atribute of the room	
	connected : boolean	//Returns in the method connectTo	
	connectedTo : List<Room>	//List to add objects of the type room	
	height : int	//Represents the height atrobite of the room	
	map : Map	//Represents the hole map	
	row : int	//Represents the row atribute of the room	
	width : int	//Represents the width atribute of the room	
	x : int	//Coordinate x	
	y : int	//Coordinate y	
Feature	passable : boolean	//Just the return to the isPassable method	
SoundManager	sounds : Clip[]	//Array of sounds from the Clip class	