Bad Philophobia

Generated by Doxygen 1.8.6

Sat Apr 12 2014 23:08:05

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

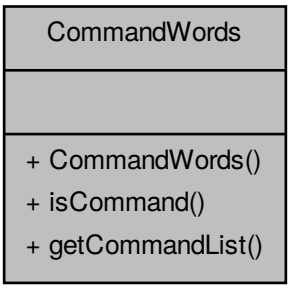
Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 CommandWords Class Reference

Collaboration diagram for CommandWords:



Public Member Functions

- CommandWords ()
- boolean isCommand (String aString)
- String getCommandList ()

4.1.1 Detailed Description

Class used to verify the commands given by the user. It contains all known commands and can verify if a String is a known command.

Author

Rémi NICOLE

Definition at line 9 of file CommandWords.java.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 CommandWords.CommandWords ( )

CommandWords class constructor.

Definition at line 21 of file CommandWords.java.

```
00021                           {
00022
00023     }
```

4.1.3 Member Function Documentation

4.1.3.1 String CommandWords.getCommandList ( )

Getter for the knownCommands field.

Definition at line 39 of file CommandWords.java.

```
00039                                   {
00040         StringBuilder commands = new StringBuilder();
00041         for(int i = 0; i < knownCommands.length; i++) {
00042             commands.append( knownCommands[i] + "   " );
00043         }
00044         return commands.toString();
00045     }
```

4.1.3.2 boolean CommandWords.isCommand (String *aString*)

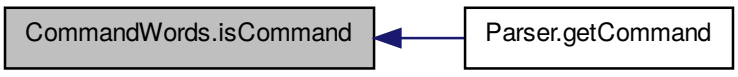
Return true if and only if the command is known.

Definition at line 28 of file CommandWords.java.

Referenced by Parser.getCommand().

```
00028                                           {
00029         for(int i = 0; i < knownCommands.length; i++) {
00030             if(knownCommands[i].equals(aString))
00031                 return true;
00032         }
00033         return false;
00034     }
```

Here is the caller graph for this function:

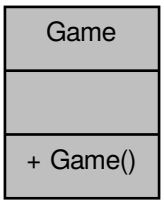


The documentation for this class was generated from the following file:

- CommandWords.java

4.2 Game Class Reference

Collaboration diagram for Game:



Public Member Functions

- Game ()

4.2.1 Detailed Description

Main class used to instantiate other objects.

Author

Rémi NICOLE

Definition at line 7 of file Game.java.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Game.Game ( )

Game class constructor

Definition at line 22 of file Game.java.

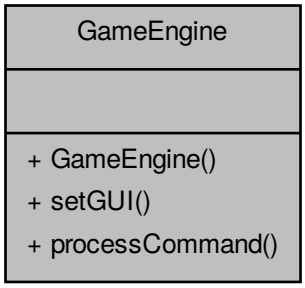
```
00022                     {
00023         engine = new GameEngine();
00024         gui = new UserInterface(engine);
00025         engine.setGUI(gui);
00026     }
```

The documentation for this class was generated from the following file:

- Game.java

4.3 GameEngine Class Reference

Collaboration diagram for GameEngine:



Public Member Functions

- GameEngine ()
- void setGUI (UserInterface userInterface)
- void processCommand (String commandLine)

4.3.1 Detailed Description

Class handling the gameplay for the game. It takes care of room, parser, and room creations and command processing.

Author

Rémi NICOLE

Definition at line 8 of file GameEngine.java.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 GameEngine.GameEngine ( )

GameEngine class constructor.

Definition at line 28 of file GameEngine.java.

```
00028                         {
00029         parser = new Parser();
00030         createRooms();
00031     }
```

4.3.3 Member Function Documentation

4.3.3.1 void GameEngine.processCommand (String *commandLine*)

Process the command.

Parameters

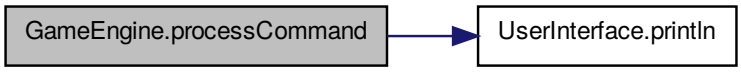
commandLine	The command to process.

Definition at line 89 of file GameEngine.java.

References UserInterface.println().

```
00089                                                {
00090         gui.println(commandLine);
00091         Command command = parser.getCommand(commandLine);
00092 
00093         if(command.isUnknown()) {
00094             gui.println("I don't know what you mean...");
00095             return;
00096         }
00097 
00098         String commandWord = command.getCommandWord();
00099         if (commandWord.equals("help"))
00100             printHelp();
00101         else if (commandWord.equals("go"))
00102             goRoom(command);
00103         else if (commandWord.equals("quit")) {
00104             if(command.hasSecondWord())
00105                 gui.println("Quit what?");
00106             else
00107                 endGame();
00108         }
00109     }
```

Here is the call graph for this function:



4.3.3.2 void GameEngine.setGUI (UserInterface *userInterface*)

Setter for the gui field.

See Also

GameEngine::gui;

Definition at line 37 of file GameEngine.java.

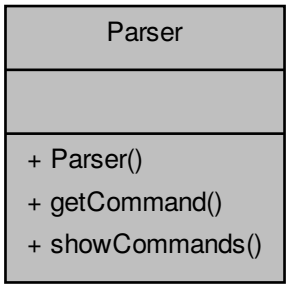
```
00037                                                {
00038         gui = userInterface;
00039         printWelcome();
00040     }
```

The documentation for this class was generated from the following file:

- GameEngine.java

4.4 Parser Class Reference

Collaboration diagram for Parser:



Public Member Functions

- Parser ()
- Command getCommand (String inputLine)
- String showCommands ()

4.4.1 Detailed Description

Class used to parse the commands with or without parameters given by the user.

Author

Rémi NICOLE

Definition at line 9 of file Parser.java.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Parser.Parser ( )

Parser class constructor.

Definition at line 19 of file Parser.java.

```
00019                     {
00020         commands = new CommandWords();
00021     }
```

4.4.3 Member Function Documentation

4.4.3.1 Command Parser.getCommand (String *inputLine*)

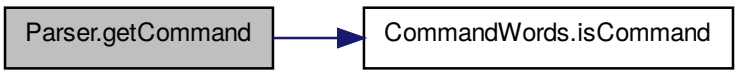
Get a new command from the user.

Definition at line 26 of file Parser.java.

References CommandWords.isCommand().

```
00026                                                {
00027
00028         String word1;
00029         String word2;
00030
00031         StringTokenizer tokenizer = new StringTokenizer(inputLine);
00032
00033         if(tokenizer.hasMoreTokens())
00034             word1 = tokenizer.nextToken();  // First word
00035         else
00036             word1 = null;
00037         if(tokenizer.hasMoreTokens())
00038             word2 = tokenizer.nextToken();  // Second word
00039         else
00040             word2 = null;
00041
00042         if(commands.isCommand(word1))
00043             return new Command(word1, word2);
00044         else
00045             return new Command(null, word2);
00046     }
```

Here is the call graph for this function:



4.4.3.2 String Parser.showCommands ( )

Getter for the knownCommands field of the commands field.

See Also

CommandWords::knownCommands

Definition at line 52 of file Parser.java.

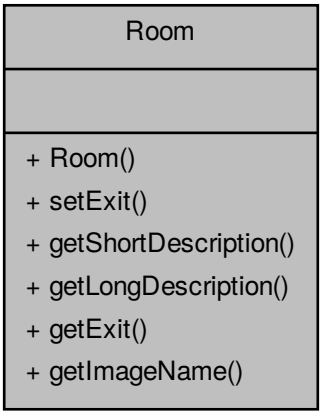
```
00052                                   {
00053         return commands.getCommandList();
00054     }
```

The documentation for this class was generated from the following file:

- Parser.java

4.5 Room Class Reference

Collaboration diagram for Room:



Public Member Functions

- Room (String description, String image)
- void setExit (String direction, Room neighbor)
- String getShortDescription ()
- String getLongDescription ()
- Room getExit (String direction)
- String getImageName ()

4.5.1 Detailed Description

Class used to handle a game's room.

Author

Rémi NICOLE

Definition at line 10 of file Room.java.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Room.Room (String *description,* String *image*)

Room class constructor.

Parameters

description	Description for the room
image	Image path to display

Definition at line 33 of file Room.java.

```
00033                                                {
00034         this.description = description;
00035         exits = new HashMap < String, Room > ();
00036         imageName = image;
00037     }
```

4.5.3 Member Function Documentation

4.5.3.1 Room Room.getExit (String *direction*)

Return the room in a specific direction.

Parameters

direction	The direction of the wanted room.

Definition at line 76 of file Room.java.

```
00076                                          {
00077         return exits.get(direction);
00078     }
```

4.5.3.2 String Room.getImageName ( )

Getter for the imageName field.

Definition at line 83 of file Room.java.

```
00083                                  {
00084         return imageName;
00085     }
```

4.5.3.3 String Room.getLongDescription ( )

Return the description of the room plus the available exits.

Definition at line 58 of file Room.java.

```
00058                                       {
00059         return "You are " + description + ".\n" + getExitString();
00060     }
```

4.5.3.4 String Room.getShortDescription ( )

Getter for the description field.

Definition at line 51 of file Room.java.

```
00051                                        {
00052         return description;
00053     }
```

4.5.3.5 void Room.setExit (String *direction,* Room *neighbor*)

Define a room in a relative direction to the current room.

Parameters

direction	Direction in which the room is.
neighbor	The room in that direction.

Definition at line 44 of file Room.java.

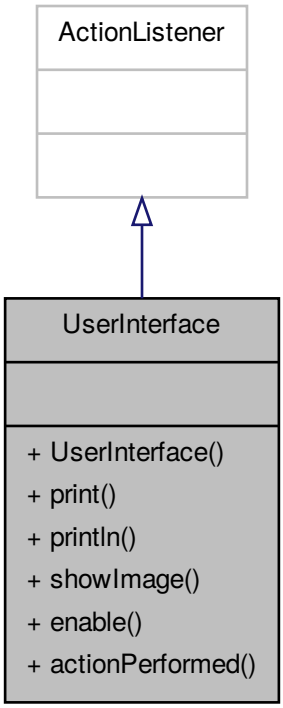
```
00044                                                   {
00045         exits.put(direction, neighbor);
00046     }
```

The documentation for this class was generated from the following file:

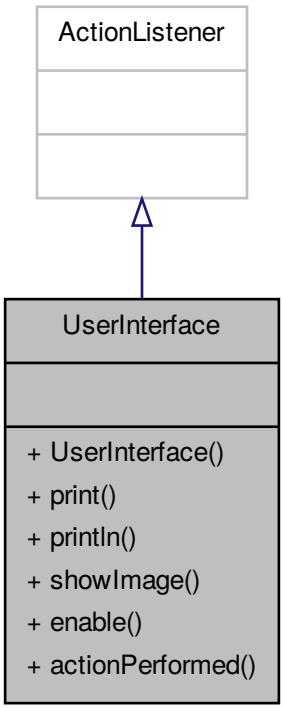
- Room.java

4.6 UserInterface Class Reference

Inheritance diagram for UserInterface:



Collaboration diagram for UserInterface:



Public Member Functions

- UserInterface (GameEngine gameEngine)
- void print (String text)
- void println (String text)
- void showImage (String imageName)
- void enable (boolean on)
- void actionPerformed (ActionEvent e)

4.6.1 Detailed Description

Class handling the game's user interface.

Author

Rémi NICOLE

Definition at line 11 of file UserInterface.java.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 UserInterface.UserInterface (GameEngine *gameEngine*)

UserInterface class constructor.

Parameters

gameEngine	The gameplay GameEngine object.

Definition at line 42 of file UserInterface.java.

```
00042                                                {
00043         engine = gameEngine;
00044         createGUI();
00045     }
```

4.6.3 Member Function Documentation

4.6.3.1 void UserInterface.actionPerformed (ActionEvent *e*)

Actionlistener for the textfield.

Parameters

e	Event.

Definition at line 129 of file UserInterface.java.

```
00129                                               {
00130         processCommand();
00131     }
```

4.6.3.2 void UserInterface.enable (boolean *on*)

Enable or disable the input field.

Definition at line 83 of file UserInterface.java.

```
00083                                    {
00084         entryField.setEditable(on);
00085         if(!on)
00086             entryField.getCaret().setBlinkRate(0);
00087     }
```

4.6.3.3 void UserInterface.print (String *text*)

Print the given text in the text area.

Parameters

text	Text to display.

Definition at line 51 of file UserInterface.java.

```
00051                                    {
00052         log.append(text);
00053         log.setCaretPosition(log.getDocument().getLength());
00054     }
```

4.6.3.4 void UserInterface.println (String *text*)

Print the given text plus a newline in the text area.

Parameters

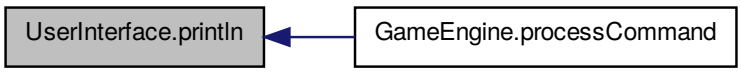
text	Text to display.

Definition at line 60 of file UserInterface.java.

Referenced by GameEngine.processCommand().

```
00060                                       {
00061         log.append(text + "\n");
00062         log.setCaretPosition(log.getDocument().getLength());
00063     }
```

Here is the caller graph for this function:



4.6.3.5 void UserInterface.showImage (String *imageName*)

Show the image corresponding to the path of the image.

Parameters

imageName	The path of the image.

Definition at line 69 of file UserInterface.java.

```
00069                                           {
00070         URL imageURL = this.getClass().getClassLoader().getResource(imageName);
00071         if(imageURL == null)
00072             System.out.println("image not found");
00073         else {
00074             ImageIcon icon = new ImageIcon(imageURL);
00075             image.setIcon(icon);
00076             myFrame.pack();
00077         }
00078     }
```

The documentation for this class was generated from the following file:

- UserInterface.java

Chapter 5

File Documentation

5.1 Command.java File Reference

Classes

- class **Command**

5.2 Command.java

```
00001
00011 class Command {
00017     private String command;
00018
00024     private String parameter;
00025
00031     public Command(String firstWord, String parameter) {
00032         command = firstWord;
00033         this.parameter = parameter;
00034     }
00035
00040     public String getCommandWord() {
00041         return command;
00042     }
00043
00048     public String getSecondWord() {
00049         return parameter;
00050     }
00051
00055     public boolean isUnknown() {
00056         return (command == null);
00057     }
00058
00062     public boolean hasSecondWord() {
00063         return (parameter != null);
00064     }
00065 }
00066
```

5.3 CommandWords.java File Reference

Classes

- class CommandWords

5.4 CommandWords.java

```
00001
```

```
00009 public class CommandWords
00010 {
00014     private static final String knownCommands[] = {
00015         "go", "quit", "help"
00016     };
00017
00021     public CommandWords() {
00022
00023     }
00024
00028     public boolean isCommand(String aString) {
00029         for(int i = 0; i < knownCommands.length; i++) {
00030             if(knownCommands[i].equals(aString))
00031                 return true;
00032         }
00033         return false;
00034     }
00035
00039     public String getCommandList() {
00040         StringBuilder commands = new StringBuilder();
00041         for(int i = 0; i < knownCommands.length; i++) {
00042             commands.append( knownCommands[i] + "   " );
00043         }
00044         return commands.toString();
00045     }
00046 }
00047
```

5.5 Game.java File Reference

Classes

- class Game

5.6 Game.java

```
00001
00007 public class Game
00008 {
00012     private UserInterface gui;
00013
00017     private GameEngine engine;
00018
00022     public Game () {
00023         engine = new GameEngine();
00024         gui = new UserInterface(engine);
00025         engine.setGUI(gui);
00026     }
00027 }
```

5.7 GameEngine.java File Reference

Classes

- class GameEngine

5.8 GameEngine.java

```
00001
00008 public class GameEngine
00009 {
00013     private Parser parser;
00014
00018     private Room currentRoom;
00019
00023     private UserInterface gui;
00024
00028     public GameEngine() {
```

```
00029         parser = new Parser();
00030         createRooms();
00031     }
00032
00037     public void setGUI(UserInterface userInterface) {
00038         gui = userInterface;
00039         printWelcome();
00040     }
00041
00045     private void printWelcome() {
00046         gui.println("Greetings human.");
00047         gui.println("I see the assassins have failed. Too bad...");
00048         gui.println("You know what they say: if you want something done, do it yourself.");
00049         gui.println("At least I can see that you don't remember anything. At last something that I can take
      advantage of.");
00050         gui.println("That was predictable, human minds are weak.");
00051         gui.println("\nBecause you're stupid, I will describe you everything that will be around us.");
00052         gui.println("Who know ? Maybe you can turn into something useful. One day. Maybe.");
00053         gui.println(currentRoom.getLongDescription());
00054         gui.showImage(currentRoom.getImageName());
00055     }
00056
00060     private void createRooms() {
00061         // create the rooms
00062         Room outside = new Room("outside the main entrance of the university", "outside.gif");
00063         Room theatre = new Room("in a lecture theatre", "castle.gif");
00064         Room pub = new Room("in the campus pub", "courtyard.gif");
00065         Room lab = new Room("in a computing lab", "stairs.gif");
00066         Room office = new Room("the computing admin office", "dungeon.gif");
00067
00068         // initialise room exits
00069         outside.setExit("east", theatre);
00070         outside.setExit("south", lab);
00071         outside.setExit("west", pub);
00072
00073         theatre.setExit("west", outside);
00074
00075         pub.setExit("east", outside);
00076
00077         lab.setExit("north", outside);
00078         lab.setExit("east", office);
00079
00080         office.setExit("west", lab);
00081
00082         currentRoom = outside;  // start game outside
00083     }
00084
00089     public void processCommand(String commandLine) {
00090         gui.println(commandLine);
00091         Command command = parser.getCommand(commandLine);
00092
00093         if(command.isUnknown()) {
00094             gui.println("I don't know what you mean...");
00095             return;
00096         }
00097
00098         String commandWord = command.getCommandWord();
00099         if (commandWord.equals("help"))
00100             printHelp();
00101         else if (commandWord.equals("go"))
00102             goRoom(command);
00103         else if (commandWord.equals("quit")) {
00104             if(command.hasSecondWord())
00105                 gui.println("Quit what?");
00106             else
00107                 endGame();
00108         }
00109     }
00110
00114     private void printHelp() {
00115         gui.println("Help ? Who needs help ? Only the weak ones.")
00116             gui.println("Your command words are: " + parser.showCommands());
00117         gui.println("That will be all.");
00118     }
00119
00125     private void goRoom(Command command) {
00126         if(!command.hasSecondWord()) {
00127             // if there is no second word, we don't know where to go...
00128             gui.println("Go where?");
00129             return;
00130         }
00131
00132         String direction = command.getSecondWord();
00133
00134         // Try to leave current room.
00135         Room nextRoom = currentRoom.getExit(direction);
00136
```

```
00137         if (nextRoom == null)
00138             gui.println("There is no door!");
00139         else {
00140             currentRoom = nextRoom;
00141             gui.println(currentRoom.getLongDescription());
00142             if(currentRoom.getImageName() != null)
00143                 gui.showImage(currentRoom.getImageName());
00144         }
00145     }
00146
00150     private void endGame() {
00151         gui.println("Thank you for playing.  Good bye.");
00152         gui.enable(false);
00153     }
00154
00155 }
```

5.9 Parser.java File Reference

Classes

- class Parser

5.10 Parser.java

```
00001 import java.util.StringTokenizer;
00002
00009 public class Parser {
00010
00014     private CommandWords commands;
00015
00019     public Parser() {
00020         commands = new CommandWords();
00021     }
00022
00026     public Command getCommand(String inputLine) {
00027
00028         String word1;
00029         String word2;
00030
00031         StringTokenizer tokenizer = new StringTokenizer(inputLine);
00032
00033         if(tokenizer.hasMoreTokens())
00034             word1 = tokenizer.nextToken();  // First word
00035         else
00036             word1 = null;
00037         if(tokenizer.hasMoreTokens())
00038             word2 = tokenizer.nextToken();  // Second word
00039         else
00040             word2 = null;
00041
00042         if(commands.isCommand(word1))
00043             return new Command(word1, word2);
00044         else
00045             return new Command(null, word2);
00046     }
00047
00052     public String showCommands() {
00053         return commands.getCommandList();
00054     }
00055 }
```

5.11 Room.java File Reference

Classes

- class Room

5.12 Room.java

```
00001 import java.util.Set;
00002 import java.util.HashMap;
00003 import java.util.Iterator;
00004
00010 public class Room {
00011
00015     private String description;
00016
00021     private HashMap < String, Room > exits;
00022
00026     private String imageName;
00027
00033     public Room(String description, String image) {
00034         this.description = description;
00035         exits = new HashMap < String, Room > ();
00036         imageName = image;
00037     }
00038
00044     public void setExit(String direction, Room neighbor) {
00045         exits.put(direction, neighbor);
00046     }
00047
00051     public String getShortDescription() {
00052         return description;
00053     }
00054
00058     public String getLongDescription() {
00059         return "You are " + description + ".\n" + getExitString();
00060     }
00061
00065     private String getExitString() {
00066         StringBuilder returnString = new StringBuilder("Exits:");
00067         for(String vS:exits.keySet())
00068             returnString.append(" " + vS);
00069         return returnString.toString();
00070     }
00071
00076     public Room getExit(String direction) {
00077         return exits.get(direction);
00078     }
00079
00083     public String getImageName() {
00084         return imageName;
00085     }
00086 }
```

5.13 UserInterface.java File Reference

Classes

- class UserInterface

5.14 UserInterface.java

```
00001 import javax.swing.*;
00002 import java.awt.*;
00003 import java.awt.event.*;
00004 import java.net.URL;
00005 import java.awt.image.*;
00006
00011 public class UserInterface implements ActionListener {
00012
00016     private GameEngine engine;
00017
00021     private JFrame myFrame;
00022
00026     private JTextField entryField;
00027
00031     private JTextArea log;
00032
00036     private JLabel image;
00037
00042     public UserInterface(GameEngine gameEngine) {
00043         engine = gameEngine;
```

```
00044         createGUI();
00045     }
00046 
00051     public void print(String text) {
00052         log.append(text);
00053         log.setCaretPosition(log.getDocument().getLength());
00054     }
00055 
00060     public void println(String text) {
00061         log.append(text + "\n");
00062         log.setCaretPosition(log.getDocument().getLength());
00063     }
00064 
00069     public void showImage(String imageName) {
00070         URL imageURL = this.getClass().getClassLoader().getResource(imageName);
00071         if(imageURL == null)
00072             System.out.println("image not found");
00073         else {
00074             ImageIcon icon = new ImageIcon(imageURL);
00075             image.setIcon(icon);
00076             myFrame.pack();
00077         }
00078     }
00079 
00083     public void enable(boolean on) {
00084         entryField.setEditable(on);
00085         if(!on)
00086             entryField.getCaret().setBlinkRate(0);
00087     }
00088 
00092     private void createGUI() {
00093         myFrame = new JFrame("Zork");
00094         entryField = new JTextField(34);
00095 
00096         log = new JTextArea();
00097         log.setEditable(false);
00098         JScrollPane listScroller = new JScrollPane(log);
00099         listScroller.setPreferredSize(new Dimension(200, 200));
00100         listScroller.setMinimumSize(new Dimension(100, 100));
00101 
00102         JPanel panel = new JPanel();
00103         image = new JLabel();
00104 
00105         panel.setLayout(new BorderLayout());
00106         panel.add(image, BorderLayout.NORTH);
00107         panel.add(listScroller, BorderLayout.CENTER);
00108         panel.add(entryField, BorderLayout.SOUTH);
00109 
00110         myFrame.getContentPane().add(panel, BorderLayout.CENTER);
00111 
00112         myFrame.addWindowListener(new WindowAdapter() {
00113             public void windowClosing(WindowEvent e) {
00114                 System.exit(0);
00115             }
00116         });
00117 
00118         entryField.addActionListener(this);
00119 
00120         myFrame.pack();
00121         myFrame.setVisible(true);
00122         entryField.requestFocus();
00123     }
00124 
00129     public void actionPerformed(ActionEvent e) {
00130         processCommand();
00131     }
00132 
00136     private void processCommand() {
00137         boolean finished = false;
00138         String input = entryField.getText();
00139         entryField.setText("");
00140 
00141         engine.interpretCommand(input);
00142     }
00143 }
```

Index