Class Game

```
java.lang.Object
  └ Game
```

```
public class Game
extends java.lang.Object
```

This class is the main class of the "World of Zuul" application. "World of Zuul" is a very simple, text based adventure game. Users can walk around some scenery. That's all. It should really be extended to make it more interesting! To play this game, create an instance of this class and call the "play" method. This main class creates and initialises all the others: it creates all rooms, creates the parser and starts the game. It also evaluates and executes the commands that the parser returns.

Version:
2006.03.30
Author:
Michael Kolling and David J. Barnes

Constructor Summary

`Game()` Create the game and initialise its internal map.

Method Summary

void	`play()` Main play routine.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Game

```
public Game()
```

Create the game and initialise its internal map.

Method Detail

play

```
public void play()
```

Main play routine. Loops until end of play.

Class Command

```
java.lang.Object
  └ Command
```

```
public class Command
extends java.lang.Object
```

This class is part of the "World of Zuul" application. "World of Zuul" is a very simple, text based adventure game. This class holds information about a command that was issued by the user. A command currently consists of two strings: a command word and a second word (for example, if the command was "take map", then the two strings obviously are "take" and "map"). The way this is used is: Commands are already checked for being valid command words. If the user entered an invalid command (a word that is not known) then the command word is . If the command had only one word, then the second word is .

Version:
2006.03.30
Author:
Michael Kolling and David J. Barnes

Constructor Summary

`Command(java.lang.String firstWord, java.lang.String secondWord)`	Create a command object.

Method Summary

`java.lang.String`	`getCommandWord()` Return the command word (the first word) of this command.
`java.lang.String`	`getSecondWord()`
`boolean`	`hasSecondWord()`
`boolean`	`isUnknown()`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Command

```
public Command(java.lang.String firstWord,
               java.lang.String secondWord)
```

Create a command object. First and second word must be supplied, but either one (or both) can be null.

Parameters:
`firstWord` - The first word of the command. Null if the command was not recognised.
`secondWord` - The second word of the command.

Method Detail

getCommandWord

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

Return the command word (the first word) of this command. If the command was not understood, the result is null.

Returns:
The command word.

getSecondWord

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

Returns:
The second word of this command. Returns null if there was no second word.

hasSecondWord

```
public boolean hasSecondWord()
```

Returns:
true if the command has a second word.

isUnknown

```
public boolean isUnknown()
```

Returns:
true if this command was not understood.

Class CommandWords

```
java.lang.Object
  └ CommandWords
```

```
public class CommandWords
extends java.lang.Object
```

This class is part of the "World of Zuul" application. "World of Zuul" is a very simple, text based adventure game. This class holds an enumeration of all command words known to the game. It is used to recognise commands as they are typed in.

Version:
2006.03.30

Author:
Michael Kolling and David J. Barnes

Constructor Summary

`CommandWords()` Constructor - initialise the command words.

Method Summary

boolean	`isCommand(java.lang.String aString)` Check whether a given String is a valid command word.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CommandWords

```
public CommandWords()
```

Constructor - initialise the command words.

Method Detail

isCommand

```
public boolean isCommand(java.lang.String aString)
```

Check whether a given String is a valid command word.

Returns:
true if a given string is a valid command, false if it isn't.

Class Parser

```
java.lang.Object
  └Parser
```

```
public class Parser
extends java.lang.Object
```

This class is part of the "World of Zuul" application. "World of Zuul" is a very simple, text based adventure game. This parser reads user input and tries to interpret it as an "Adventure" command. Every time it is called it reads a line from the terminal and tries to interpret the line as a two word command. It returns the command as an object of class Command. The parser has a set of known command words. It checks user input against the known commands, and if the input is not one of the known commands, it returns a command object that is marked as an unknown command.

Version:
2006.03.30
Author:
Michael Kolling and David J. Barnes

Constructor Summary

`Parser()` Create a parser to read from the terminal window.

Method Summary

`Command`	`getCommand()`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Parser

```
public Parser()
```

Create a parser to read from the terminal window.

Method Detail

getCommand

```
public Command getCommand()
```

Returns:
The next command from the user.

Class Room

```
java.lang.Object
  └ Room
```

```
public class Room
extends java.lang.Object
```

Class Room - a room in an adventure game. This class is part of the "World of Zuul" application. "World of Zuul" is a very simple, text based adventure game. A "Room" represents one location in the scenery of the game. It is connected to other rooms via exits. The exits are labelled north, east, south, west. For each direction, the room stores a reference to the neighboring room, or null if there is no exit in that direction.

Version:
2006.03.30
Author:
Michael Kolling and David J. Barnes

Field Summary

java.lang.String	**description**
Room	**eastExit**
Room	**northExit**
Room	**southExit**
Room	**westExit**

Constructor Summary

Room(java.lang.String description) Create a room described "description".

Method Summary

java.lang.String	**getDescription**()
void	**setExits**(Room north, Room east, Room south, Room west) Define the exits of this room.

Methods inherited from class java.lang.Object

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

Field Detail

description

```
public java.lang.String description
```

eastExit

```
public Room eastExit
```

northExit

```
public Room northExit
```

southExit

```
public Room southExit
```

westExit

```
public Room westExit
```

Constructor Detail

Room

```
public Room(java.lang.String description)
```

Create a room described "description". Initially, it has no exits. "description" is something like "a kitchen" or "an open court yard".

Parameters:
description - The room's description.

Method Detail

getDescription

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

Returns:
The description of the room.

setExits

```
public void setExits(Room north,
                     Room east,
                     Room south,
                     Room west)
```

Define the exits of this room. Every direction either leads to another room or is null (no exit there).

Parameters:

`north` - The north exit.
`east` - The east east.
`south` - The south exit.
`west` - The west exit.
