

```
/**
 * 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.
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (February 2002)
 */
```

```
public class Game
{
    private Parser parser;
    private Room currentRoom;

    /**
     * Create the game and initialise its internal map.
     */
    public Game()
    {
        createRooms();
        parser = new Parser();
    }

    /**
     * Create all the rooms and link their exits together.
     */
    private void createRooms()
    {
        Room outside, theatre, pub, lab, office;
        // create the rooms
        outside = new Room("outside the main entrance of the university");
        theatre = new Room("in a lecture theatre");
        pub = new Room("in the campus pub");
        lab = new Room("in a computing lab");
        office = new Room("in the computing admin office");
        // initialise room exits
        outside.setExits(null, theatre, lab, pub);
        theatre.setExits(null, null, null, outside);
        pub.setExits(null, outside, null, null);
        lab.setExits(outside, office, null, null);
        office.setExits(null, null, null, lab);

        currentRoom = outside; // start game outside
    }

    /**
     * Main play routine. Loops until end of play.
     */
}
```

```
public void play()
{
    printWelcome();
    // Main command loop. Repeatedly read and execute commands
    boolean finished = false;
    while (! finished) {
        Command command = parser.getCommand();
        finished = processCommand(command);
    }
    System.out.println("Thank you for playing. Good bye.");
}

/**
 * Print out the opening message for the player.
 */
private void printWelcome()
{
    System.out.println("Welcome to the World of Zuul! World of Zuul");
    System.out.println("is a new, incredibly boring adventure game");
    System.out.println("Type 'help' if you need help.");
    System.out.println("You are " + currentRoom.getDescription());
    System.out.print("Exits: ");
    if(currentRoom.northExit != null)
        System.out.print("north ");
    if(currentRoom.eastExit != null)
        System.out.print("east ");
    if(currentRoom.southExit != null)
        System.out.print("south ");
    if(currentRoom.westExit != null)
        System.out.print("west ");
    System.out.println();
}

/**
 * Given a command, process (execute) the command. If this command
 * ends the game, true is returned, otherwise false is returned.
 */
private boolean processCommand(Command command)
{
    boolean wantToQuit = false;
    if(command.isUnknown()) {
        System.out.println("I don't know what you mean...");
        return false;
    }
    String commandWord = command.getCommandWord();
    if (commandWord.equals("help"))
        printHelp();
    else if (commandWord.equals("go"))
        goRoom(command);
    else if (commandWord.equals("quit"))
        wantToQuit = quit(command);
    return wantToQuit;
}
```

```
/**
 * Print out some help information. Here we print some
 * stupid, cryptic message and a list of the command words.
 */
private void printHelp()
{
    System.out.println("You are lost. You are alone. You wander");
    System.out.println("around at the university.");
    System.out.println("Your command words are:");
    System.out.println("    go quit help");
}

/**
 * Try to go to one direction. If there is an exit, enter
 * the new room, otherwise print an error message.
 */
private void goRoom(Command command)
{
    if(!command.hasSecondWord()) {
        // if there is no second word, we don't know where to go...
        System.out.println("Go where?");
        return;
    }
    String direction = command.getSecondWord();
    // Try to leave current room.
    Room nextRoom = null;
    if(direction.equals("north"))
        nextRoom = currentRoom.northExit;
    if(direction.equals("east"))
        nextRoom = currentRoom.eastExit;
    if(direction.equals("south"))
        nextRoom = currentRoom.southExit;
    if(direction.equals("west"))
        nextRoom = currentRoom.westExit;

    if (nextRoom == null)
        System.out.println("There is no door!");
    else {
        currentRoom = nextRoom;
        System.out.println("You are " + currentRoom.getDescription());
        System.out.print("Exits: ");
        if(currentRoom.northExit != null)
            System.out.print("north ");
        if(currentRoom.eastExit != null)
            System.out.print("east ");
        if(currentRoom.southExit != null)
            System.out.print("south ");
        if(currentRoom.westExit != null)
            System.out.print("west ");
        System.out.println();
    }
}
}
```

```
/*
 * 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.
 * @author Michael Kolling and David J. Barnes
 */

public class Room
{
    public String description;
    public Room northExit;
    public Room southExit;
    public Room eastExit;
    public Room westExit;

    /**
     * Create a room described "description". Initially, it has no exits.
     * "description" is something like "a kitchen" or "an open court yard"
     */
    public Room(String description)
    {
        this.description = description;
    }

    /**
     * Define the exits of this room. Every direction either leads
     * to another room or is null (no exit there).
     */
    public void setExits(Room north, Room east, Room south, Room west)
    {
        if(north != null)
            northExit = north;
        if(east != null)
            eastExit = east;
        if(south != null)
            southExit = south;
        if(west != null)
            westExit = west;
    }

    /**
     * Return the description of the room (the one that was defined
     * in the constructor).
     */
    public String getDescription()
    {
        return description;
    }
}
```