NCC Level-5DC Diploma in Computing

Analysis, Design and Implementation

Candidates Name	**: Fatema Akter**
ID No	**: 00154713**
Module Title	**: Analysis, Design and Implementation**
Assignment Title	**: Text Adventures**
Examination Cycle:	**September 2015**

Candidate attempting to gain an unfair advantage or colluding in anyway whatsoever (other than on joint assignments) are liable to be disqualified. Plagiarism is an offence.

Expected candidate time allocation: 35 to 40 hours

Mark	Moderated Mark	Final Mark

Marker's comment:

Moderator's comment:

Statement of Confirmation of Own Work

Programmed /qualification name:

Student Declaration:

I have read and understood the NCC Education's policy on Academic Dishonesty and Plagiarism.

I can confirm the following details:

Student ID/Registration number	: **00154713**
Name	: **Fatema Akter**
Center Name	: **Daffodil Institute of Information Technology.**
Module Name	: **Analysis, Design and Implementation**
Assignment Title	: **Text Adventures**
Number of Words	: **348**

I confirm that this is my own and that I have not plagiarized any part of it. I have also noted the assessment criteria and pass mark for assignments.

Due Date:

Submitted Date:

Student Signature:

Table of Contents

Introduction

I am given an assignment to create a system which will allow for simple version of a text adventure game to be created. And the assignment has five tasks such as, candidate class list and diagrams, activity diagram, use case diagram, and code architecture and system implementations. I have tried my best to fulfill all the requirement of the assignment.

Task-1

Candidate class list and diagrams:

Class list:

There are many kinds of class in this assignment. I have selected some class such as-

- Player
- Parser
- Rooms
- Descriptions
- Items
- Commands
- Inventory

Class diagram:

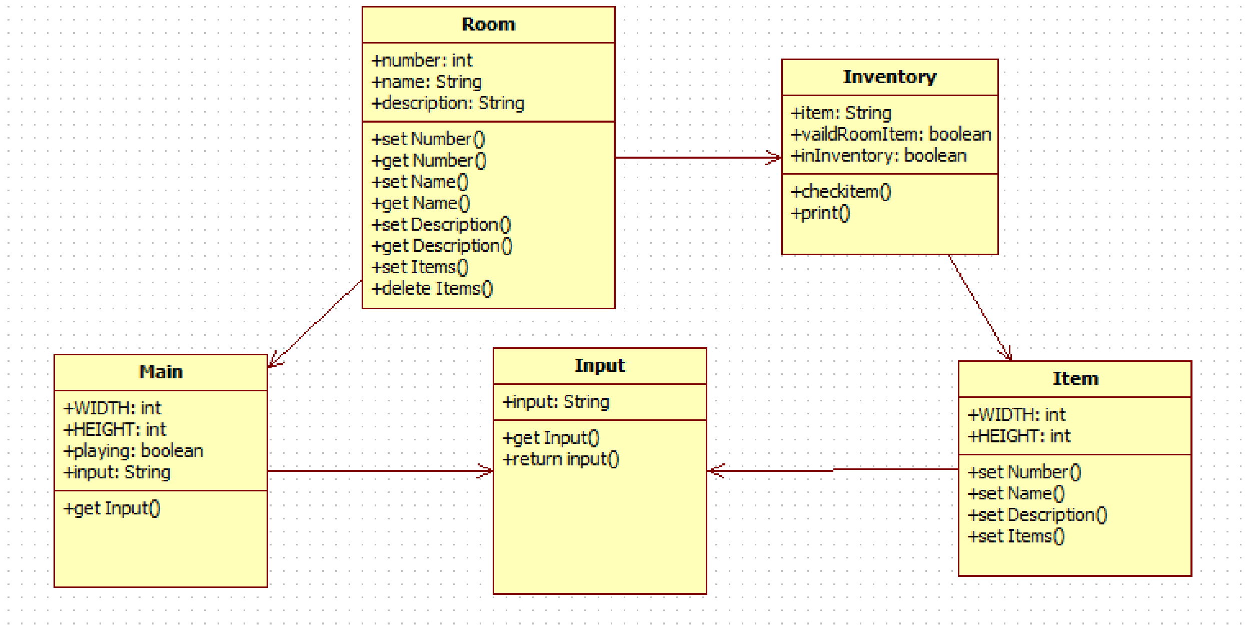


Figure No-1.1: Diagram of Class

(Anon., n.d.)

Task-2

Activity diagram:

Diagram for Main java activity:

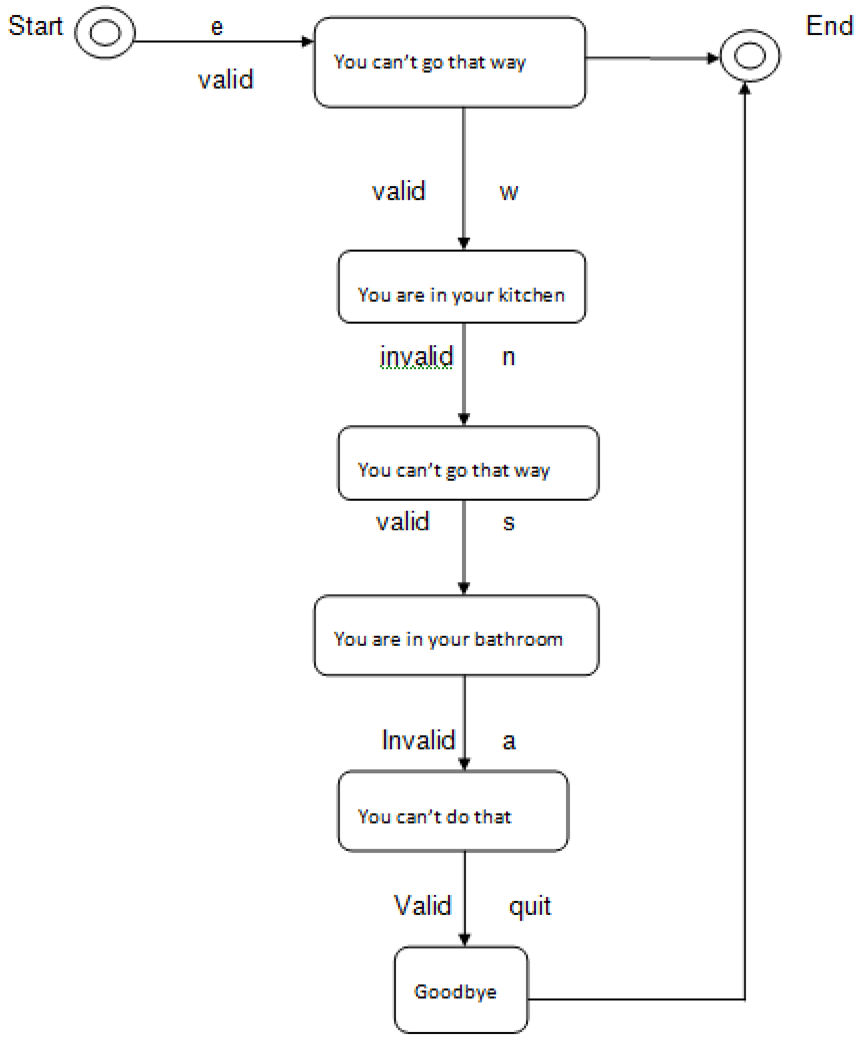


Figure No-2.1: Activity Diagram of Main class

Diagram for item java activity:

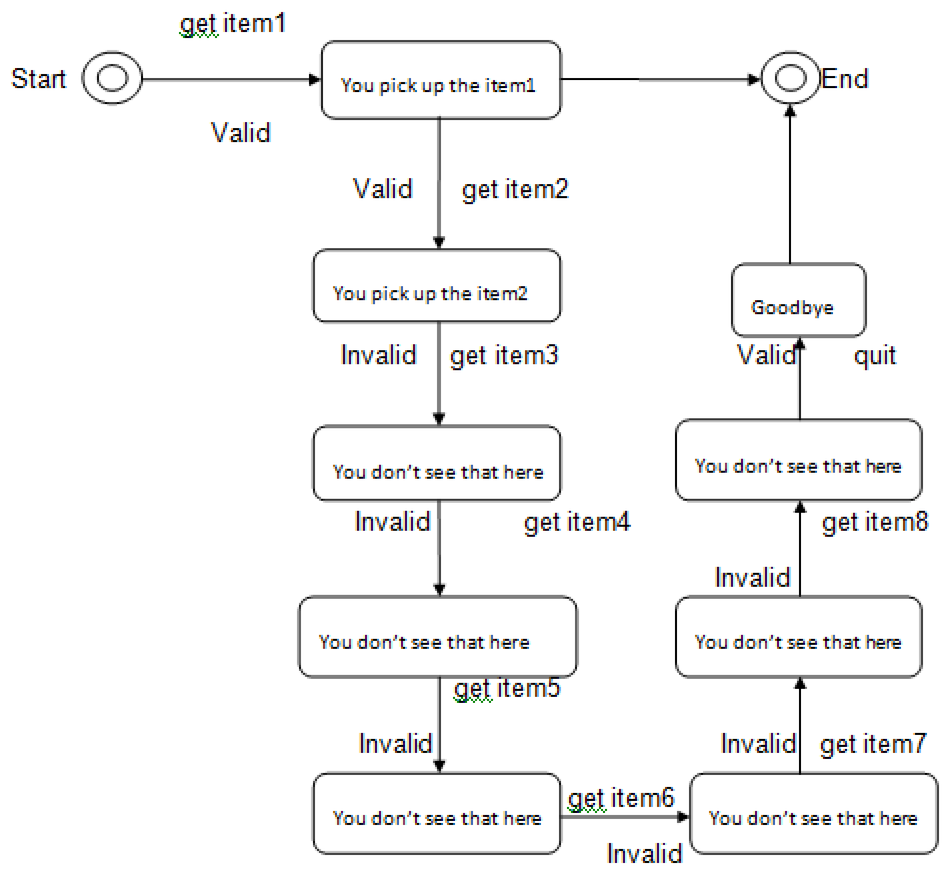


Figure No-2.2: Activity Diagram of item class

Diagram for inventory java activity:

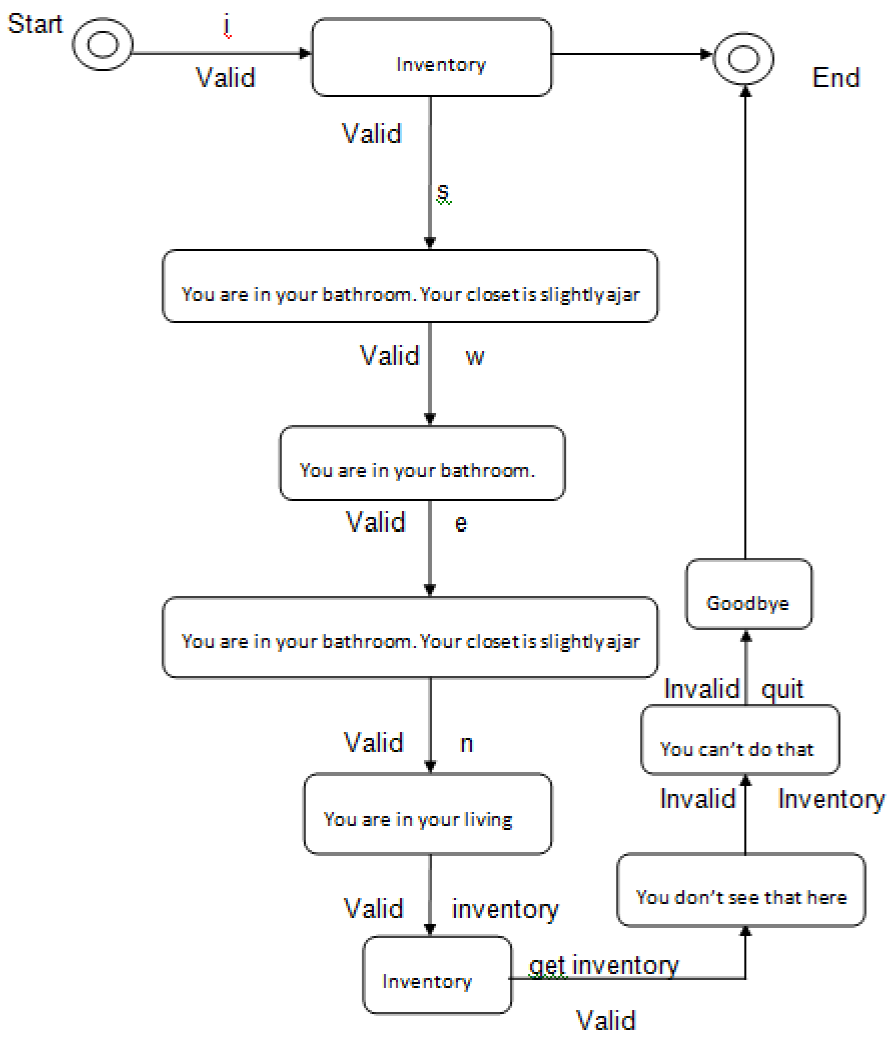


Figure No-2.3: Activity Diagram of Inventory class

(Anon., n.d.)

Task-3

Use case diagram:

Diagram for player:

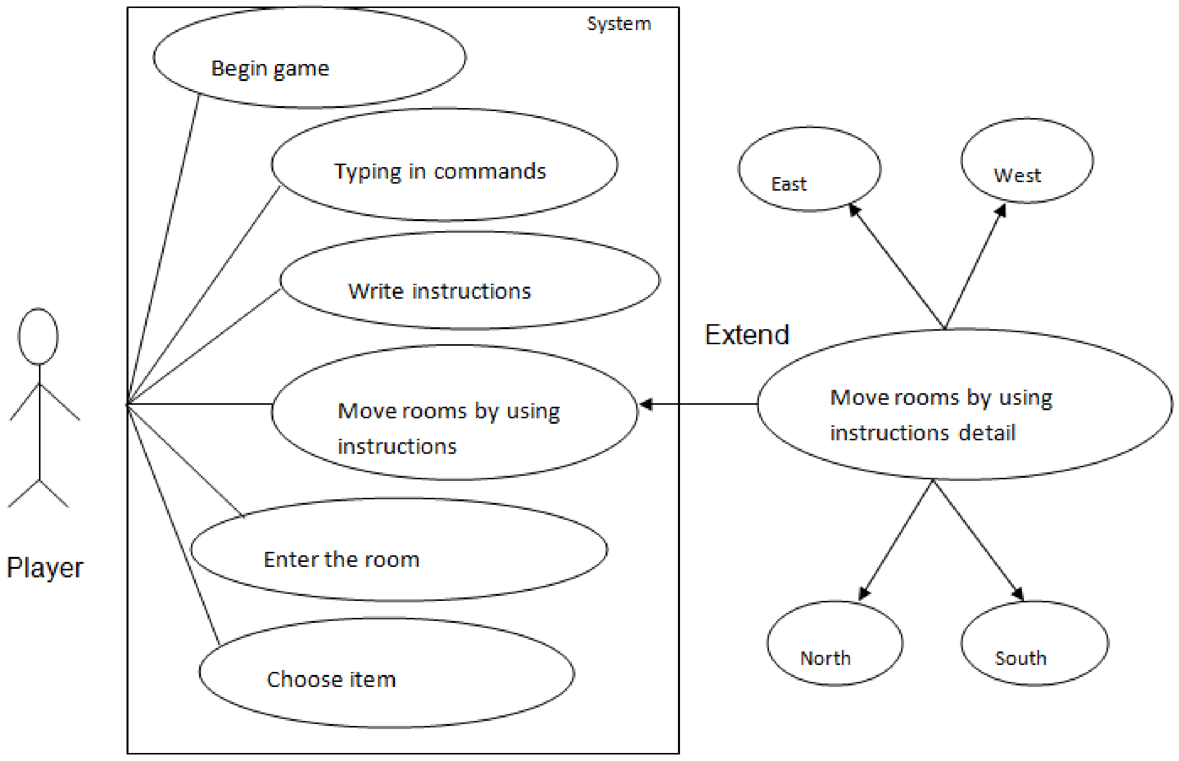


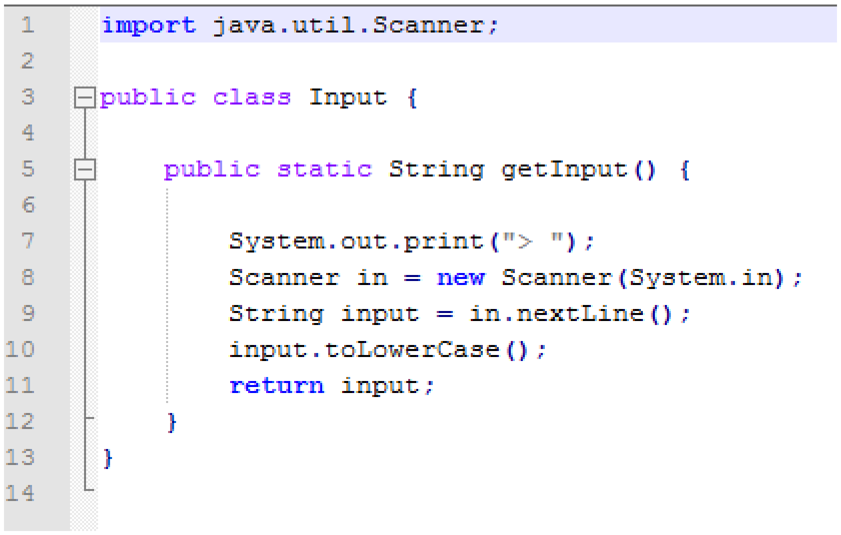
Figure No-3.1: Use Case Diagram of Player

(Education, 2011)

Task-4

Code architecture:

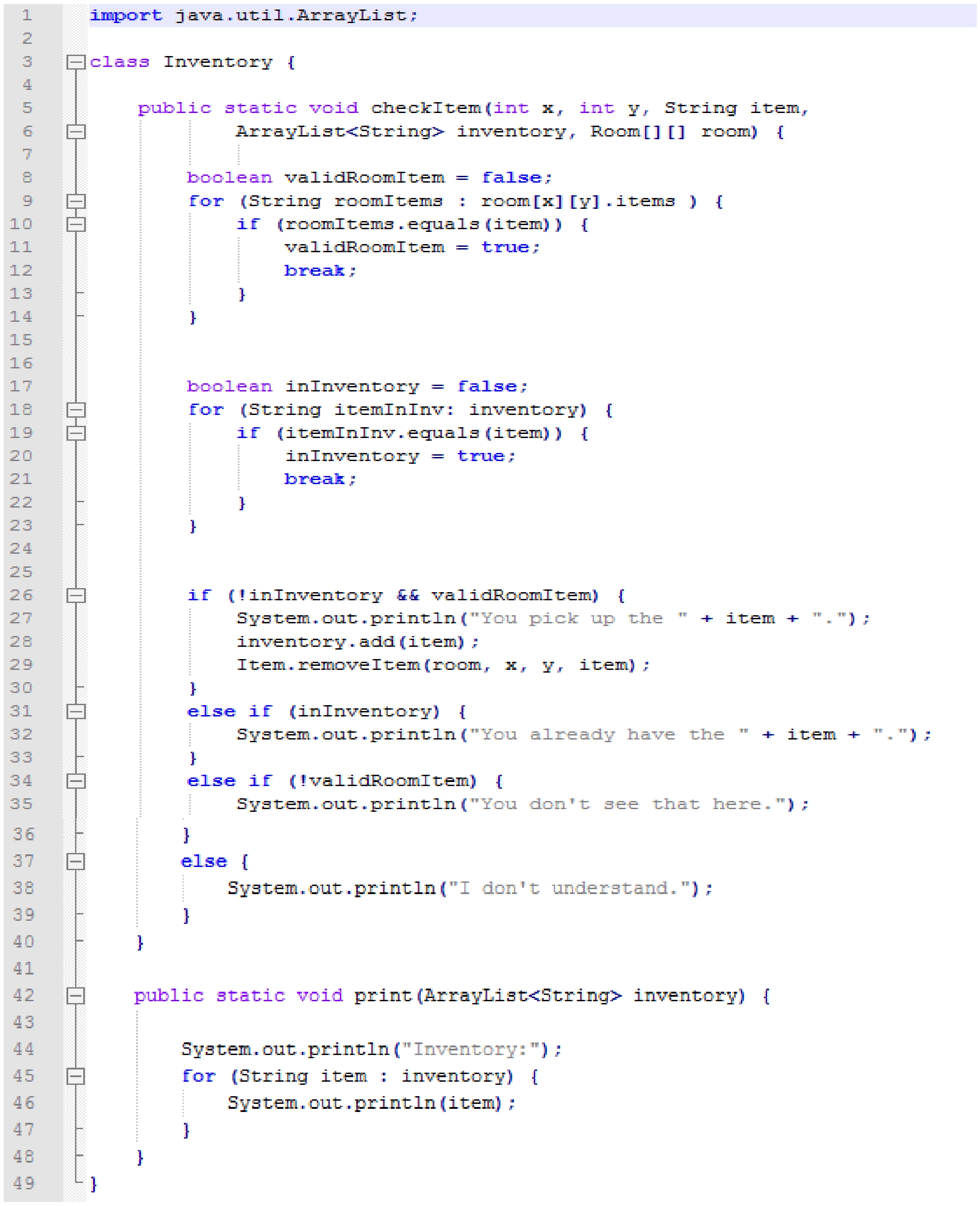
Code for input java:



```java
import java.util.Scanner;

public class Input {

    public static String getInput() {

        System.out.print("> ");
        Scanner in = new Scanner(System.in);
        String input = in.nextLine();
        input.toLowerCase();
        return input;
    }
}

```

Figure No-4.1: Architecture code of input java

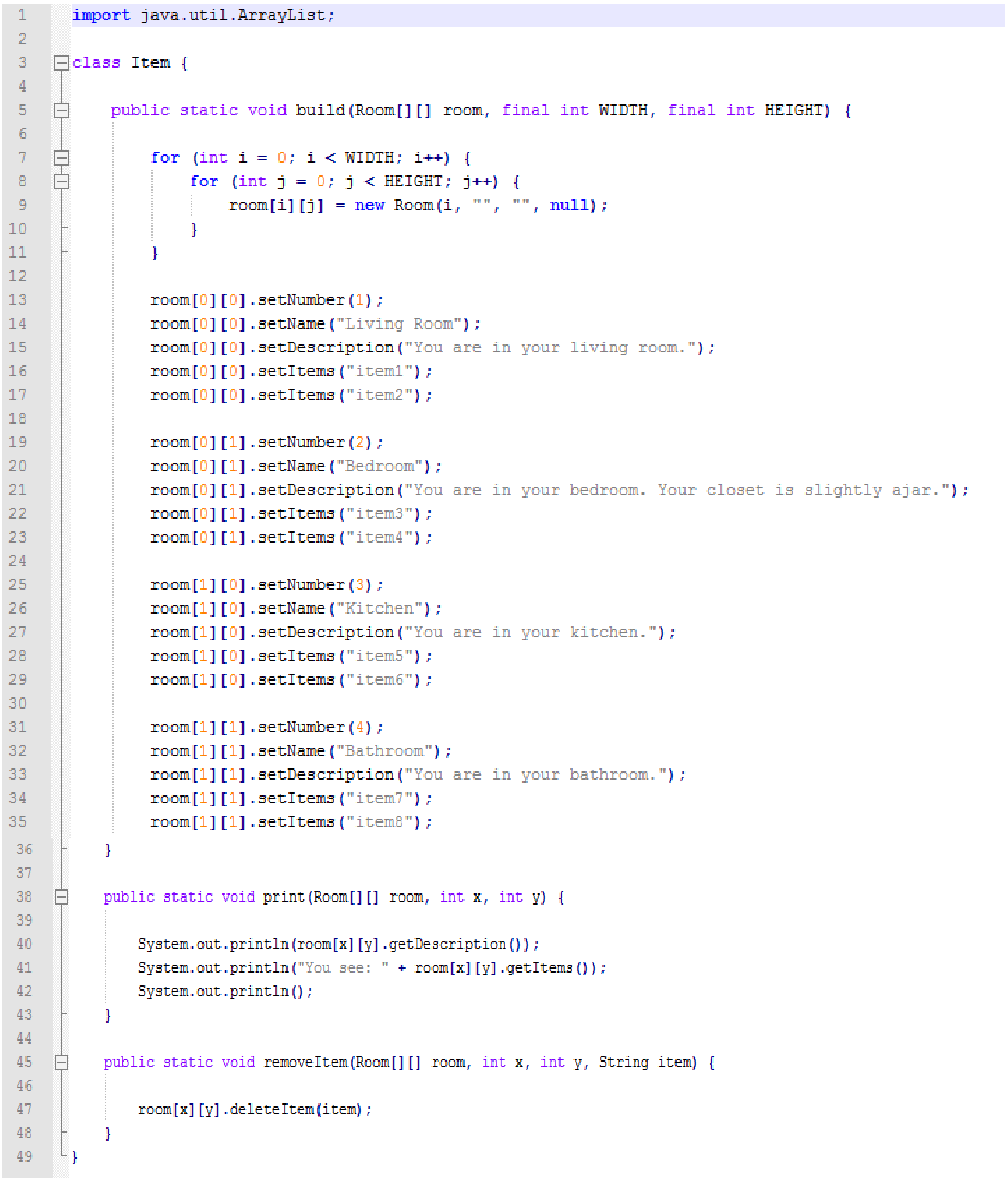
Code for Inventory java:



```java
import java.util.ArrayList;

class Inventory {

    public static void checkItem(int x, int y, String item,
            ArrayList<String> inventory, Room[][] room) {

        boolean validRoomItem = false;
        for (String roomItems : room[x][y].items ) {
            if (roomItems.equals(item)) {
                validRoomItem = true;
                break;
            }
        }


        boolean inInventory = false;
        for (String itemInInv: inventory) {
            if (itemInInv.equals(item)) {
                inInventory = true;
                break;
            }
        }


        if (!inInventory && validRoomItem) {
            System.out.println("You pick up the " + item + ".");
            inventory.add(item);
            Item.removeItem(room, x, y, item);
        }
        else if (inInventory) {
            System.out.println("You already have the " + item + ".");
        }
        else if (!validRoomItem) {
            System.out.println("You don't see that here.");
        }
        else {
            System.out.println("I don't understand.");
        }
    }

    public static void print(ArrayList<String> inventory) {

        System.out.println("Inventory:");
        for (String item : inventory) {
            System.out.println(item);
        }
    }
}
```

Figure No-4.2: Architecture code of inventory java

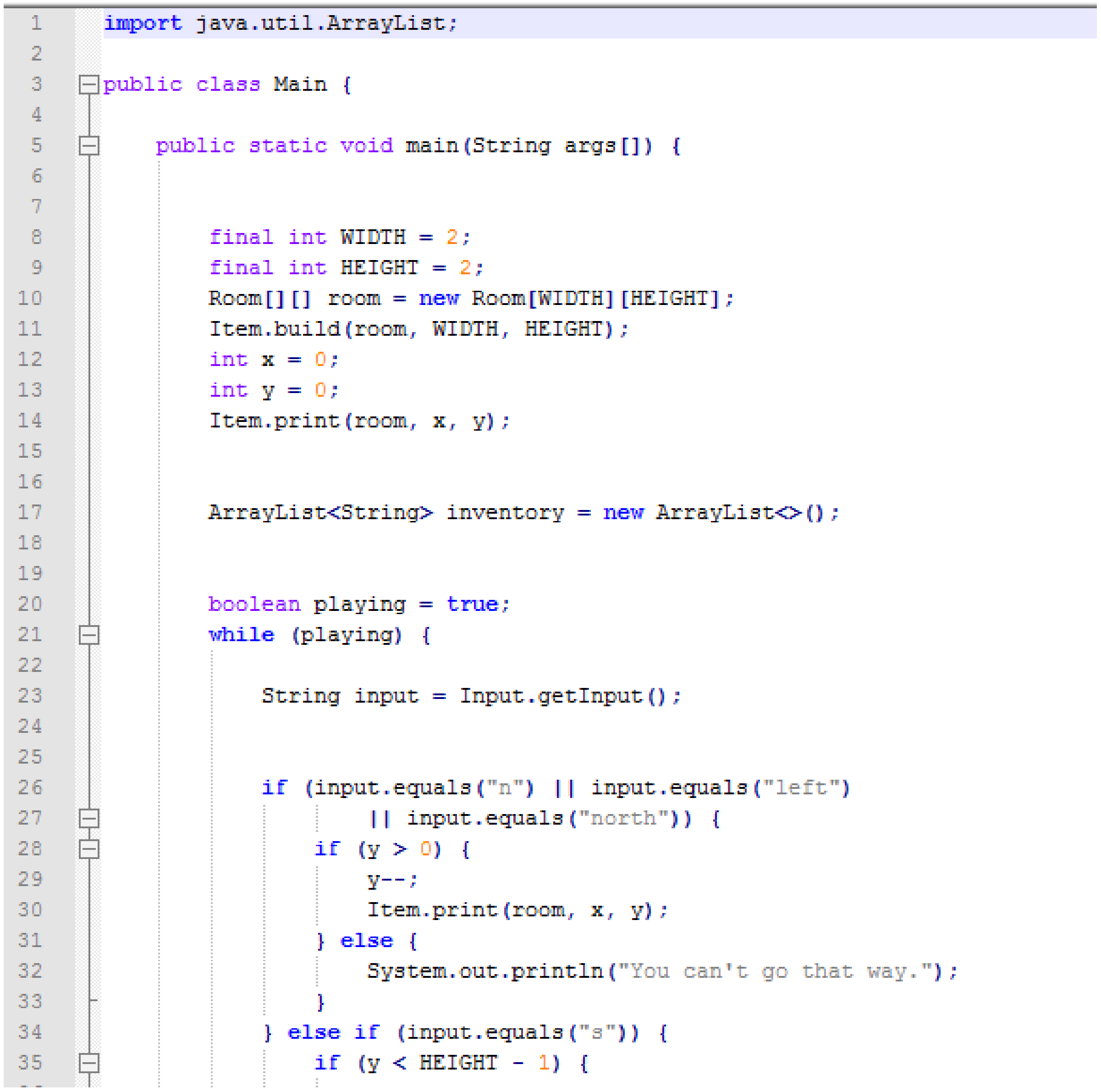
Code for item java:



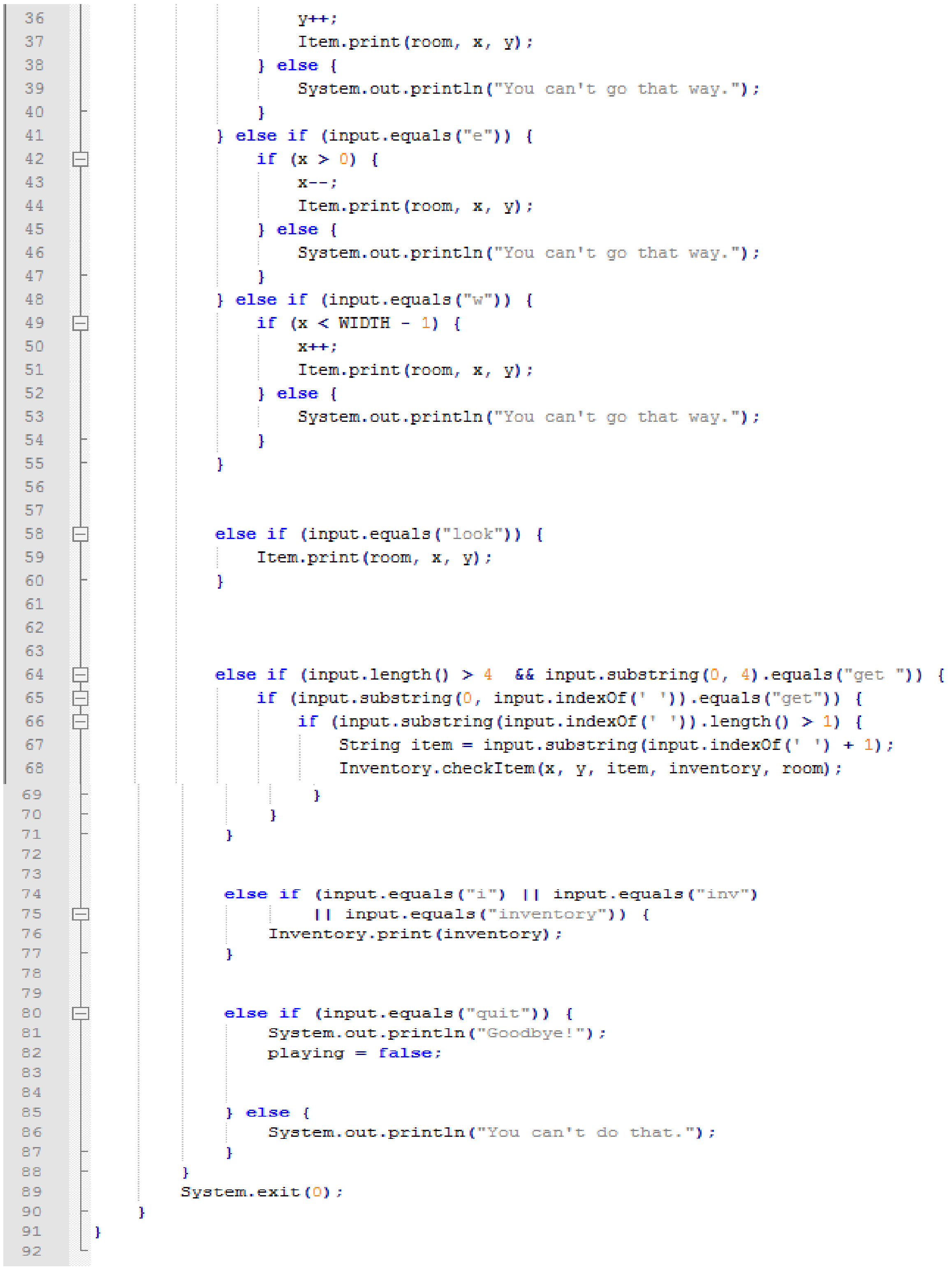
```java
import java.util.ArrayList;

class Item {

    public static void build(Room[][] room, final int WIDTH, final int HEIGHT) {

        for (int i = 0; i < WIDTH; i++) {
            for (int j = 0; j < HEIGHT; j++) {
                room[i][j] = new Room(i, "", "", null);
            }
        }

        room[0][0].setNumber(1);
        room[0][0].setName("Living Room");
        room[0][0].setDescription("You are in your living room.");
        room[0][0].setItems("item1");
        room[0][0].setItems("item2");

        room[0][1].setNumber(2);
        room[0][1].setName("Bedroom");
        room[0][1].setDescription("You are in your bedroom. Your closet is slightly ajar.");
        room[0][1].setItems("item3");
        room[0][1].setItems("item4");

        room[1][0].setNumber(3);
        room[1][0].setName("Kitchen");
        room[1][0].setDescription("You are in your kitchen.");
        room[1][0].setItems("item5");
        room[1][0].setItems("item6");

        room[1][1].setNumber(4);
        room[1][1].setName("Bathroom");
        room[1][1].setDescription("You are in your bathroom.");
        room[1][1].setItems("item7");
        room[1][1].setItems("item8");
    }

    public static void print(Room[][] room, int x, int y) {

        System.out.println(room[x][y].getDescription());
        System.out.println("You see: " + room[x][y].getItems());
        System.out.println();
    }

    public static void removeItem(Room[][] room, int x, int y, String item) {

        room[x][y].deleteItem(item);
    }
}
```

Figure No-4.3: Architecture code of item java

Code for Main java:



```java
import java.util.ArrayList;

public class Main {

    public static void main(String args[]) {


        final int WIDTH = 2;
        final int HEIGHT = 2;
        Room[][] room = new Room[WIDTH][HEIGHT];
        Item.build(room, WIDTH, HEIGHT);
        int x = 0;
        int y = 0;
        Item.print(room, x, y);


        ArrayList<String> inventory = new ArrayList<>();


        boolean playing = true;
        while (playing) {

            String input = Input.getInput();


            if (input.equals("n") || input.equals("left")
                    || input.equals("north")) {
                if (y > 0) {
                    y--;
                    Item.print(room, x, y);
                } else {
                    System.out.println("You can't go that way.");
                }
            } else if (input.equals("s")) {
                if (y < HEIGHT - 1) {
```

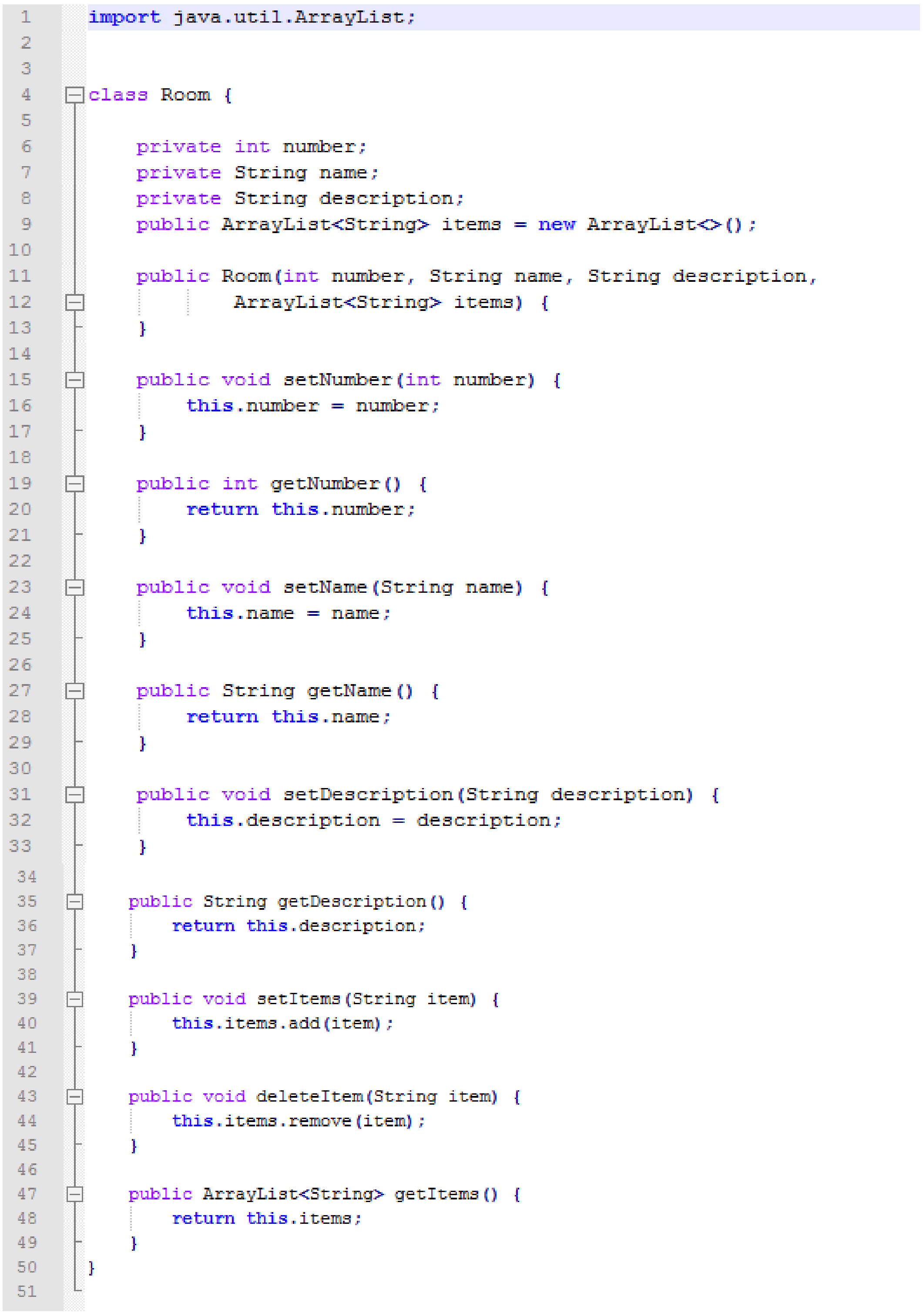


```
                    y++;
                    Item.print(room, x, y);
                } else {
                    System.out.println("You can't go that way.");
                }
            } else if (input.equals("e")) {
                if (x > 0) {
                    x--;
                    Item.print(room, x, y);
                } else {
                    System.out.println("You can't go that way.");
                }
            } else if (input.equals("w")) {
                if (x < WIDTH - 1) {
                    x++;
                    Item.print(room, x, y);
                } else {
                    System.out.println("You can't go that way.");
                }
            }


            else if (input.equals("look")) {
                Item.print(room, x, y);
            }


            else if (input.length() > 4  && input.substring(0, 4).equals("get ")) {
                if (input.substring(0, input.indexOf(' ')).equals("get")) {
                    if (input.substring(input.indexOf(' ')).length() > 1) {
                        String item = input.substring(input.indexOf(' ') + 1);
                        Inventory.checkItem(x, y, item, inventory, room);
                    }
                }
            }


            else if (input.equals("i") || input.equals("inv")
                    || input.equals("inventory")) {
                Inventory.print(inventory);
            }


            else if (input.equals("quit")) {
                System.out.println("Goodbye!");
                playing = false;


            } else {
                System.out.println("You can't do that.");
            }
        }
        System.exit(0);
    }
}
```

Figure No-4.4: Architecture code of Main java

(Anon., n.d.)

Code for Room java:

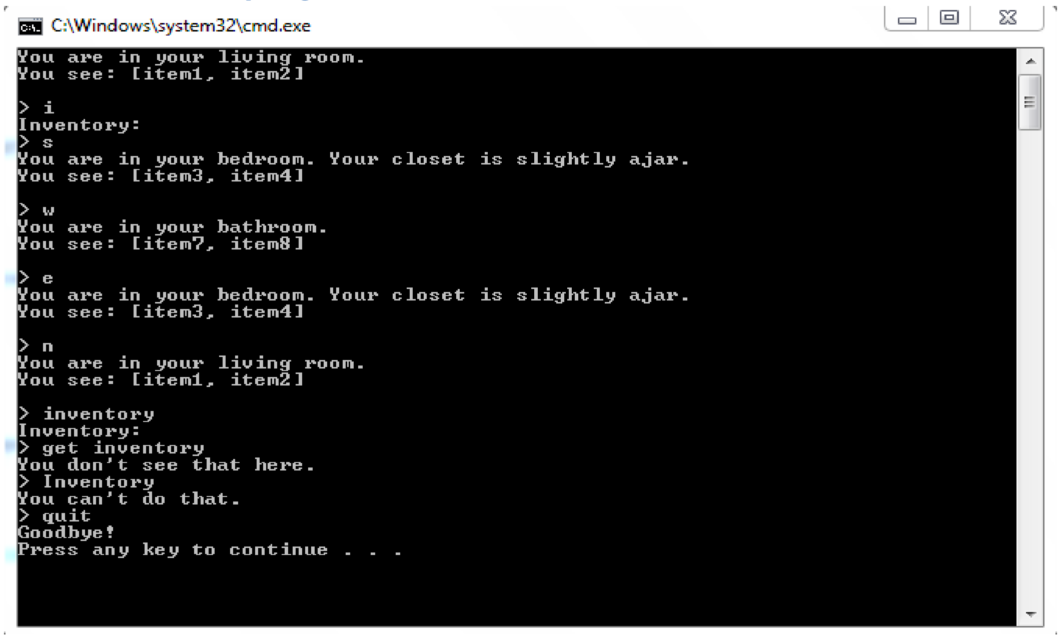


```java
import java.util.ArrayList;


class Room {

    private int number;
    private String name;
    private String description;
    public ArrayList<String> items = new ArrayList<>();

    public Room(int number, String name, String description,
            ArrayList<String> items) {
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public int getNumber() {
        return this.number;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getDescription() {
        return this.description;
    }

    public void setItems(String item) {
        this.items.add(item);
    }

    public void deleteItem(String item) {
        this.items.remove(item);
    }

    public ArrayList<String> getItems() {
        return this.items;
    }
}
```

Figure No-4.5: Architecture code of Room java

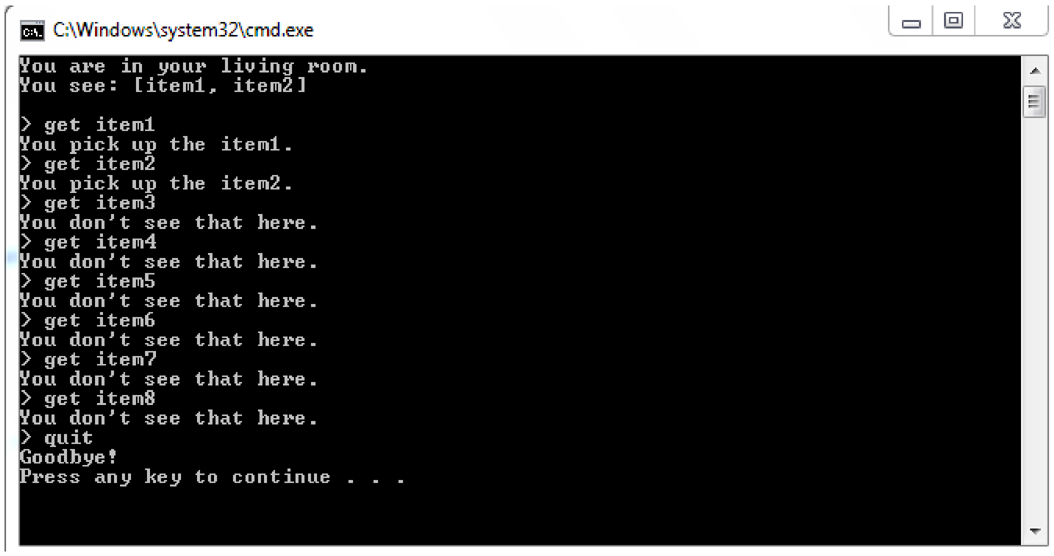
(Anon., n.d.)

Output Result of Java program:



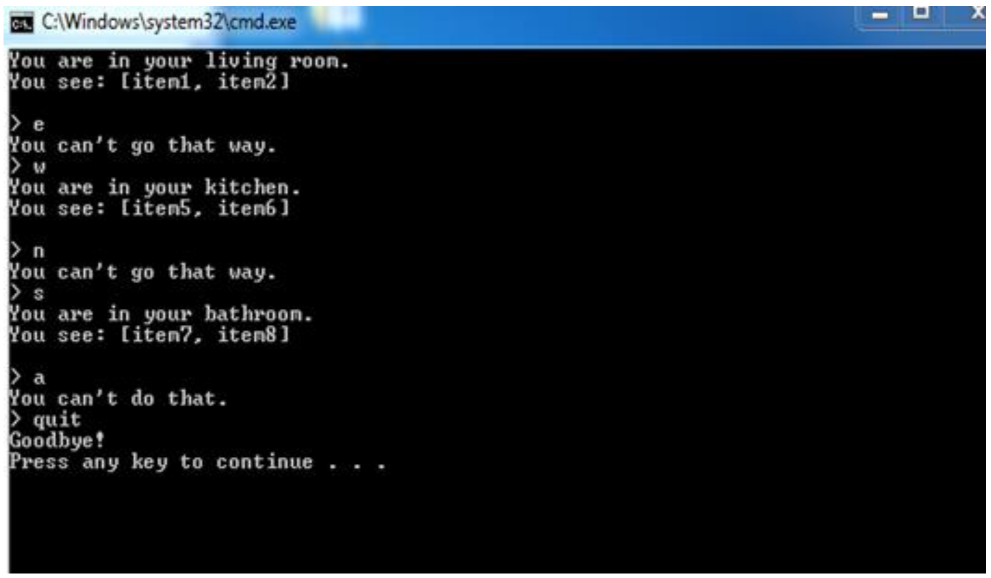
```
You are in your living room.
You see: [item1, item2]

> i
Inventory:
> s
You are in your bedroom. Your closet is slightly ajar.
You see: [item3, item4]

> w
You are in your bathroom.
You see: [item7, item8]

> e
You are in your bedroom. Your closet is slightly ajar.
You see: [item3, item4]

> n
You are in your living room.
You see: [item1, item2]

> inventory
Inventory:
> get inventory
You don't see that here.
> Inventory
You can't do that.
> quit
Goodbye!
Press any key to continue . . .
```

Figure No-4.6: output of main java program



```
You are in your living room.
You see: [item1, item2]

> get item1
You pick up the item1.
> get item2
You pick up the item2.
> get item3
You don't see that here.
> get item4
You don't see that here.
> get item5
You don't see that here.
> get item6
You don't see that here.
> get item7
You don't see that here.
> get item8
You don't see that here.
> quit
Goodbye!
Press any key to continue . . .
```

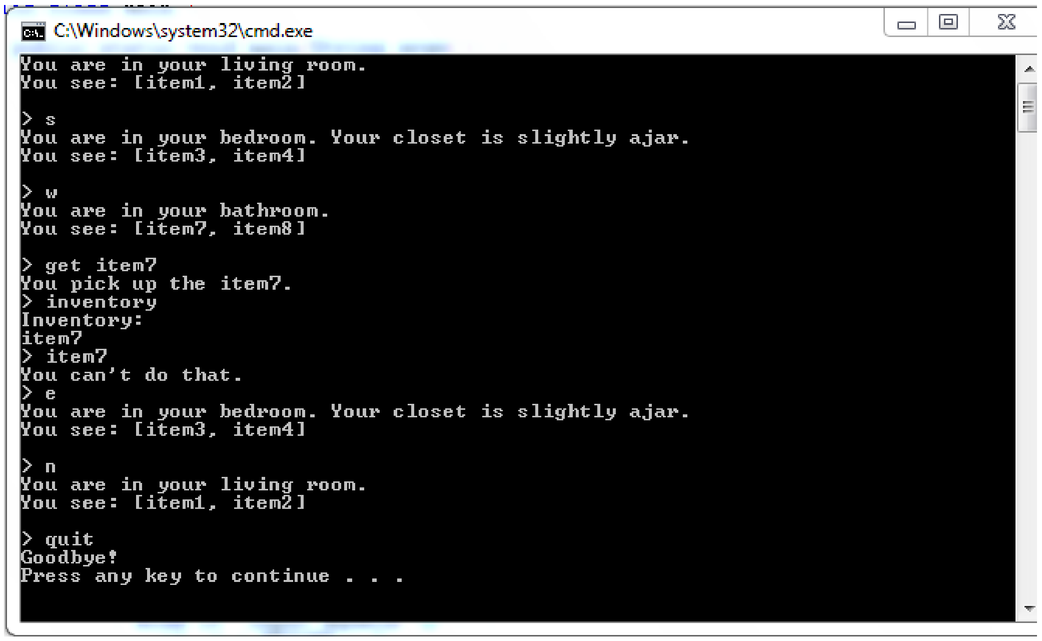
Figure No-4.7: output of main java program



C:\Windows\system32\cmd.exe

```
You are in your living room.
You see: [item1, item2]

> e
You can't go that way.
> w
You are in your kitchen.
You see: [item5, item6]

> n
You can't go that way.
> s
You are in your bathroom.
You see: [item7, item8]

> a
You can't do that.
> quit
Goodbye!
Press any key to continue . . .
```

Figure No-4.8: output of main java program



C:\Windows\system32\cmd.exe

```
You are in your living room.
You see: [item1, item2]

> s
You are in your bedroom. Your closet is slightly ajar.
You see: [item3, item4]

> w
You are in your bathroom.
You see: [item7, item8]

> get item7
You pick up the item7.
> inventory
Inventory:
item7
> item7
You can't do that.
> e
You are in your bedroom. Your closet is slightly ajar.
You see: [item3, item4]

> n
You are in your living room.
You see: [item1, item2]

> quit
Goodbye!
Press any key to continue . . .
```

Figure No-4.9: output of main java program

Task-5

System implementations:

My program runs successfully and I have attached my program in a CD with the assignment.

Conclusion:

In the achievement of this assignment I had a great working experience. As I was able to create a java program with the help Text pad .I also had a good learning about start UML which was applied for creating UML diagram. I got to learn about activity diagram. From this experience I am hoping to develop and design better java program in the future for further analysis.

Bibliography

Anon., 2015. [Online] Available at: http://staruml.sourceforge.net.

Anon., n.d. [Online] Available at: http://staruml.sourceforge.net [Accessed 10 july 2015].

Anon., n.d. [Online] Available at: http://staruml.sourceforge.net [Accessed 10 july 2015].

Anon., n.d. [Online] Available at: http://www.monkeys-at-keyboards.com/java2/31.shtml [Accessed 11 july 2015].

Anon., n.d. [Online] Available at: http://basildoncoder.com/blog/2008/03/21/the-pg-wodehouse-method-of-refactoring/ [Accessed 12 july 2015].

Anon., n.d. [Online] Available at: http://www.soberit.hut.fi/mmantyla/BadCodeSmellsTaxonomy.htm [Accessed 13 july 2015].

Education, N., ed., 2011. Analysis, Design and Implementation Student Guide.