PF-Semester Project

Instructor: Sir Azfar Shakeel Khan

Functional Contact & Cell Management

Prepared By:

Fa20-Bse-094 (M.Ruslan Babar)
Fa20-Bse-072 (Sarah Nasir)
Fa20-Bse-057 (Muneeb Arshad)
Fa20-Bse-059 (M.Saad Iqbal)
Fa20-Bse-083 (Warda Rana)

Date: 28th May 2021

Project Contribution:

01 (Fa20-Bse-094) M.Ruslan Babar:

- Class: Functional Contact & Cell Management
  - Main method (Control Transfering)
- Class Contact Book:
  - Sign up for contact book.
  - Log in to contact book.
  - →Successful Login mechanism

02 (Fa20-Bse-057) Muneeb Arshad:

- Class Contact Book:
  - Mechanism after successful login to contact book:
  - Adding contact data to file.
  - Reading contact Data from file.
  - Save contact Data to file

03 (Fa20-Bse-072) Sarah Nasir:

- Class Contact Book:
  - Mechanism after successful login to contact book:
  - Finding contact Data from file.
  - Deleting contact data from file.
  - Updating contact data in a file.

04 (Fa20-Bse-059) M.Saad Iqbal:

- Class Gamification-Arena:
  - Main method of gamification-arena.
  - Method Hangman Game().
  - Modularization for Random word generation.
  - User Game Play method.

05 (Fa20-Bse-083) Warda Rana:

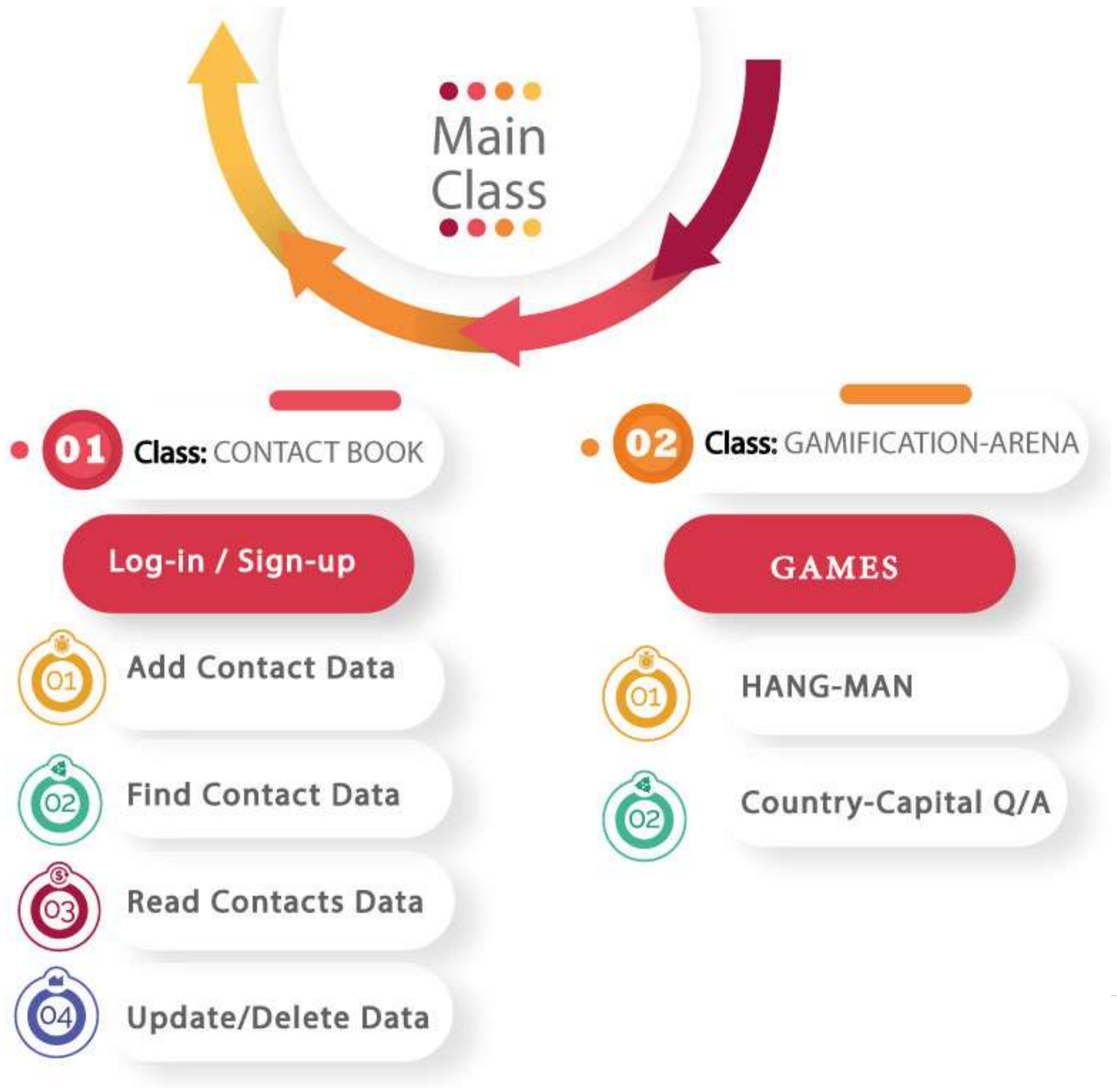
- Class Gamification-Arena:
  - Main method of gamification-arena.
  - Method Country Capital Game().
  - Modularization for Shuffling random word.
  - Final Game Summery.

1. Introduction:

A Java project that has a title **Functional Contact and Cell Management.** Primarily, the purpose of this project is to store actual contact data permanently to a file. That can be used in any form any time and has features to add, delete, update, search etc. Any user can make their own contact book profile but firstly he has to sign-up for an account. After sign-up, contact book profile is accessible to the **registered user**s. This features are handled by **File Handling** techniques. This project also covers the gamification phase that has several technical games that literally covers the concepts of Selections and Control Structures, Multi-Dimensional Arrays more efficiently. Moreover, the **flow** is comfortable, the user can access any method/function any time by following the instructions on display.

2. Structure:

This is the basic structure of our Project to create a bold image in readers mind



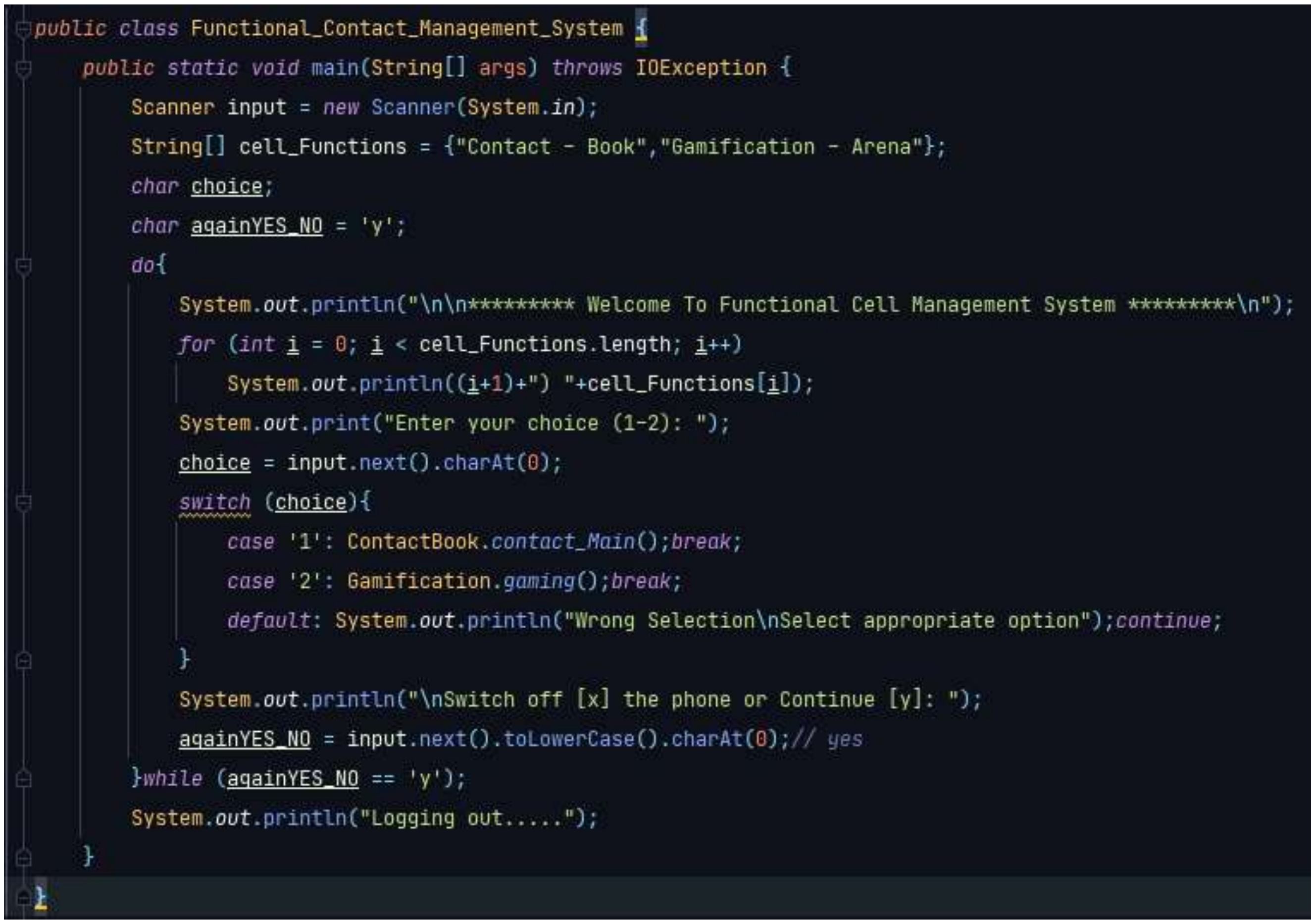
Project: Functional Contact & Cell Management.

1. Main Class (Functional Contact & Cell Management):

It's the **brain** class of the project consists of a single method which is the main method. It hold all the data required for project startup. A user based input is required to continue the flow of program and repeated asked the user in case of wrong input. Following point to be kept in mind:

- A **switch statement** is used to control the user based input.
- **Do while** loop is used to continuously ask the user in case of wrong input as well as using the program again.
- Furthermore the do while **loop** has an invoking statements, that invoke other classes and their method, according to the user.
- If the user presses invalid key, then **try catch block** will catch the **error** and asks again to enter a proper input.

Snap Main Class



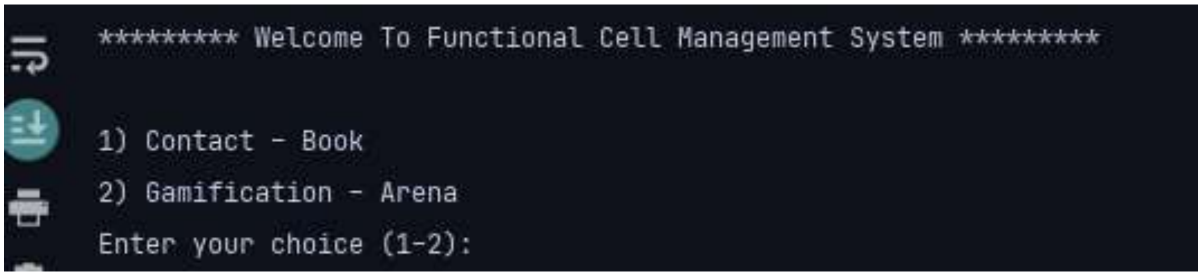
```
public class Functional_Contact_Management_System {
    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);
        String[] cell_Functions = {"Contact - Book","Gamification - Arena"};
        char choice;
        char againYES_NO = 'y';
        do{
            System.out.println("\n\n********* Welcome To Functional Cell Management System *********\n");
            for (int i = 0; i < cell_Functions.length; i++)
                System.out.println((i+1)+") "+cell_Functions[i]);
            System.out.print("Enter your choice (1-2): ");
            choice = input.next().charAt(0);
            switch (choice){
                case '1': ContactBook.contact_Main();break;
                case '2': Gamification.gaming();break;
                default: System.out.println("Wrong Selection\nSelect appropriate option");continue;
            }
            System.out.println("\nSwitch off [x] the phone or Continue [y]: ");
            againYES_NO = input.next().toLowerCase().charAt(0);// yes
        }while (againYES_NO == 'y');
        System.out.println("Logging out.....");
    }
}
```

3.1 Invoking from main class:

Since this project cover the two Phases of a cell phone therefore, the main public class can invoke two other classes and their methods according to user. The other two class are given below:

- Contact-Book.
- Gamification-Arena.

Snap:



```
********* Welcome To Functional Cell Management System *********

1) Contact - Book
2) Gamification - Arena
Enter your choice (1-2):
```

Options:

- Pressing 1 invokes the method **ContactBook** ()
- Pressing 2 invokes the method **Gamification_Arena** ()

Class 1: Contact-Book:

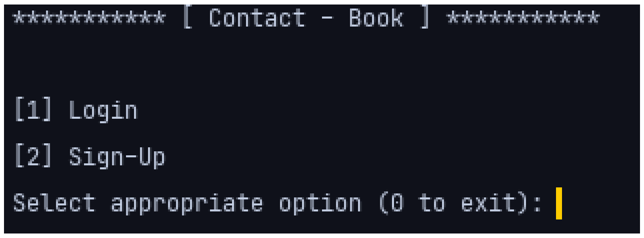
After selecting Contact-Book choice, you will be redirected to credentials page. Before going into the detailed working of contact book, you must kept in mind the two options

- Sign-up.
- Log-in.

Whenever a user accesses this class, he prompted with the above two options. For continuous and smooth flow **do while** and **switch statement** is used, which also helped in invoking the (Login & Signup methods). In case of wrong input the Control structure executes again.

- Pressing 1 invokes the method **Login_page** ()
- Pressing 2 invokes the method **Signup_Page**()

Snap:



```
************ [ Contact - Book ] ************


[1] Login
[2] Sign-Up
Select appropriate option (0 to exit):
```

Concrete and reusable data on disk:

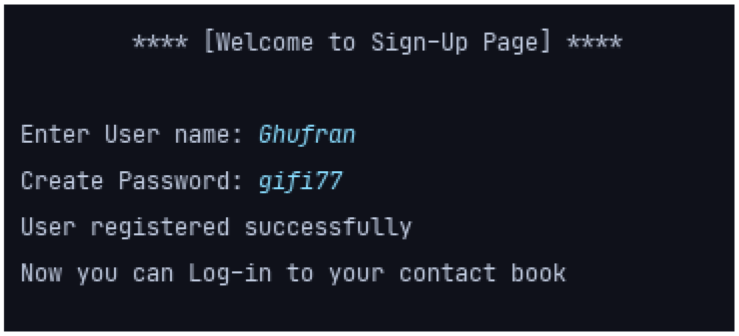
Contact-Book consistently uses the feature of **File-Handling** thought the class. This allows the registered users to store contact data on disk that can be used in future. Every **registered** user can read, write, delete, update and find a specific contact data from file.

Sign-up: (method)

A new user must sign up for the contact book. During sign, user is asked a username and password. After providing details text file is created **(Registration Area .txt)** using **createNewFile() method**, that hold the user credentials throughout the program even after the program termination. There credentials which consists of username and password can be used for future use. A user cannot log-in unless he is registered or signed-up. Following are the Sign-up features.

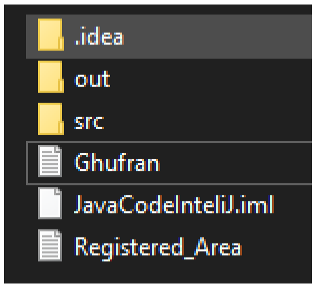
- A new user can't use contact book. Sign-up is necessary.
- During Sign-up, user credentials are secured in a file called (Registration Area.txt). These credentials are used for log-in purposes later.
- After successful sign-up, a contact book file is generated that is accessible to the user who signed-up.
- Now the user has access to his own contact book. And has the ability to use contact book any time.

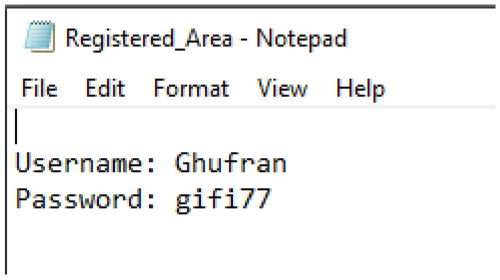
Snap: Sign-up



```
        **** [Welcome to Sign-Up Page] ****

Enter User name: Ghufran
Create Password: gifi77
User registered successfully
Now you can Log-in to your contact book
```

Registration File:





```
Username: Ghufran
Password: gifi77
```

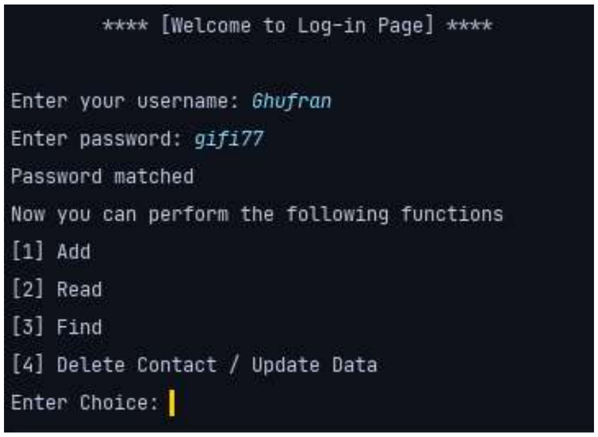
Description:

- In above figure **Ghufran** refer to the contact book and the user who signed up. Now he has got a contact book and can add, delete, find, update data to his Contact book.
- **Registered_Area** is a text file which holds all the credentials data of existing or new users. Each time a user tries to log-in to his contact book, this file is read and the user verification is checked.

Log-in: (method)

A registered user is welcomed to log-in to contact book. This login-panel ask the username and password from the user. When the user enters the username and password, the compiler will read that text file (**Registration_Area.txt**). If the user entered **credentials** matches the credentials in text file then the user can enter his work space of contact book. If it doesn't matches the credentials then the compiler prompted with a message **(Log-in Failed. Try other username or sign-up)**. This means that only registered users can log-in and Sign-up is necessary for new users. After successful login, a method named successful login is called for further actions:

Snap:



```
        **** [Welcome to Log-in Page] ****

Enter your username: Ghufran
Enter password: gifi77
Password matched
Now you can perform the following functions
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice:
```

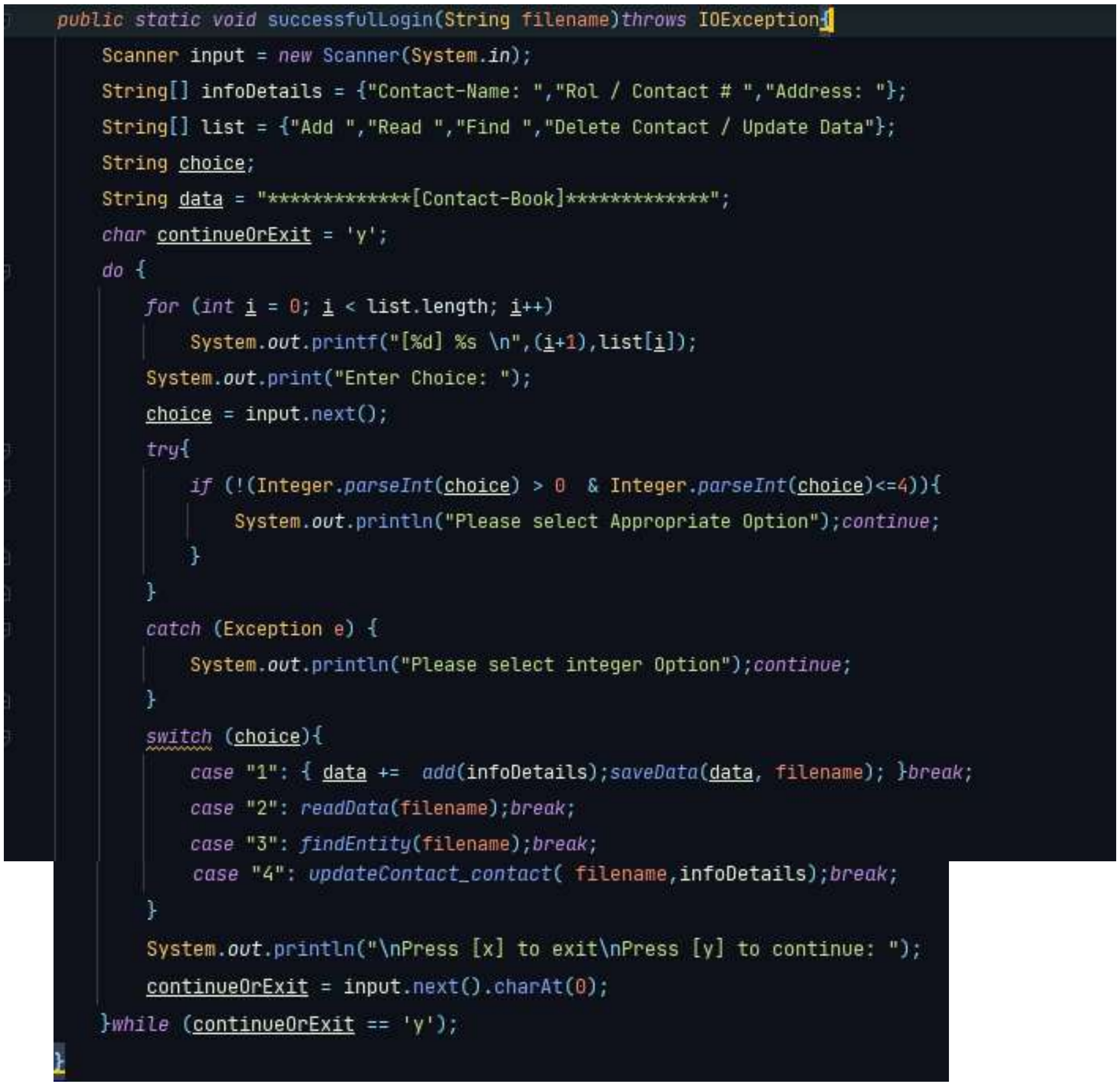
Successful Log-in:

After entering this method, a logged-in *(registered user)* has access to the following accessible features.

- Add Contact Data.
- Read Contact Data.
- Find Contact Data.
- Delete / Updated Contact Data.

In order to access these features, the user is prompted with the above options and asked for an integer input. A user based choice is controlled by switch statement and In case of wrong input the Control structure (**do while**) is used to executes again unless a proper input in given.

Snap Code



```
public static void successfulLogin(String filename)throws IOException{
    Scanner input = new Scanner(System.in);
    String[] infoDetails = {"Contact-Name: ","Rol / Contact # ","Address: "};
    String[] list = {"Add ","Read ","Find ","Delete Contact / Update Data"};
    String choice;
    String data = "*************[Contact-Book]*************";
    char continueOrExit = 'y';
    do {
        for (int i = 0; i < list.length; i++)
            System.out.printf("[%d] %s \n",(i+1),list[i]);
        System.out.print("Enter Choice: ");
        choice = input.next();
        try{
            if (!(Integer.parseInt(choice) > 0  & Integer.parseInt(choice)<=4)){
                System.out.println("Please select Appropriate Option");continue;
            }
        }
        catch (Exception e) {
            System.out.println("Please select integer Option");continue;
        }
        switch (choice){
            case "1": { data +=  add(infoDetails);saveData(data, filename); }break;
            case "2": readData(filename);break;
            case "3": findEntity(filename);break;
            case "4": updateContact_contact( filename,infoDetails);break;
        }
        System.out.println("\nPress [x] to exit\nPress [y] to continue: ");
        continueOrExit = input.next().charAt(0);
    }while (continueOrExit == 'y');
}
```

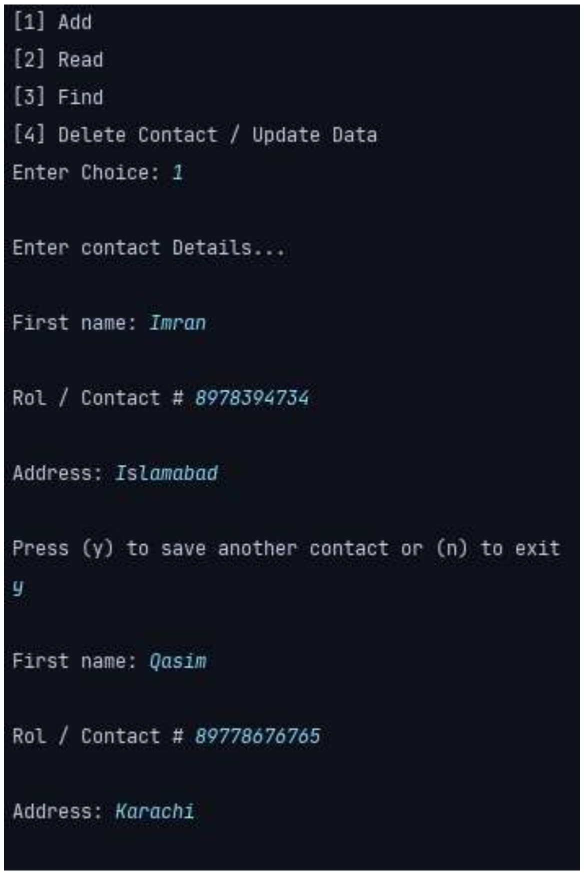
• (Feature) Add Contact Data:

This is a method of Contact Book derived from the method (successful login). A logged in (Registered user) can add data to his own contact book file being created in disk during Sign-up process. Now when the user enters this method, he is prompted with a message to enter contact data.

- Contact Name:
- Roll / Contact # details:
- Contact Address details

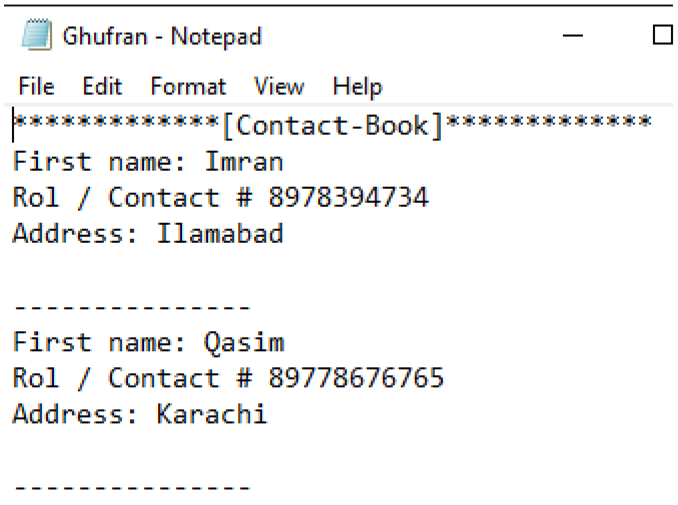
Snap:

[1] Add:



```
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice: 1

Enter contact Details...

First name: Imran

Rol / Contact # 8978394734

Address: Islamabad

Press (y) to save another contact or (n) to exit
y

First name: Qasim

Rol / Contact # 89778676765

Address: Karachi
```

Contact-Book: Ghufran



```
*************[Contact-Book]*************
First name: Imran
Rol / Contact # 8978394734
Address: Ilamabad

---------------
First name: Qasim
Rol / Contact # 89778676765
Address: Karachi

---------------
```

After Adding two contacts, the data is stored on the disk in text file that can be used later for searching, updating, deleting and adding etc.

- Whenever user enters data, that is written to a contact book which is a text file.

- This data is stored on disk, and can be used in future for updating, deletion and searching etc.
- Any user can make his Contact Book, but he has to be signed up initially.

After entering the relevant data. The user is asked again if he wants to save another contact details. When the user finishes data entry, this data is transferred to a method called (**saveData** method) that writes the data onto the file that is located on disk. And this is done using file handling techniques.

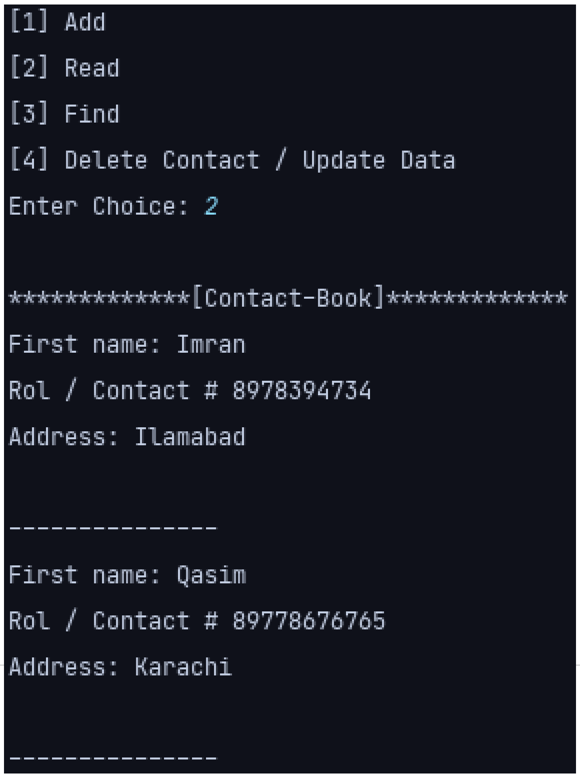
- Save Data method:
  This method involves the basic functionality of file handling. This basic feature is **File Writing**. The data being **transferred** from Add Data method is written to file in this method. File writer object is created for writing. In order to verify the data being written, we can locate the file from the disk and verify the data being written.

(Feature) Read Contact data:

This is a method of Contact Book derived from the method (successful login). A logged in (Registered user) can read data to his own contact book. For file reading Scanner Object and some methods like **(scan.hasNext())** and **.next()** are used. After accessing this feature following possibilities can occur:

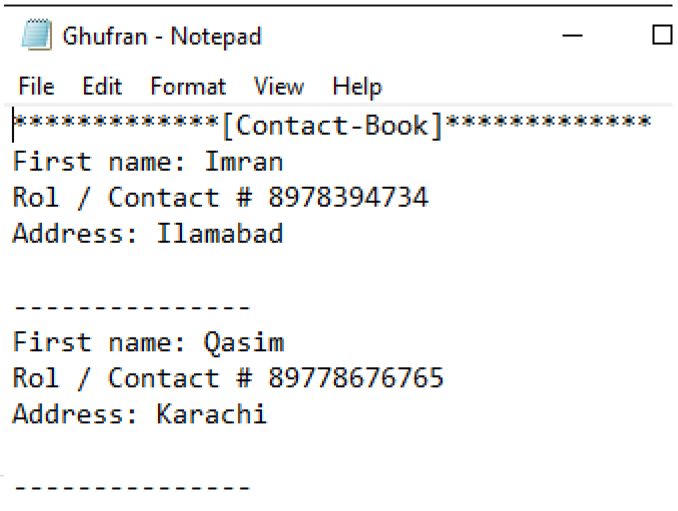
- If the contact book exits and contains data, then the whole data is read from file and displayed on terminal using while loop and **hasNext()** method.
- If file does not exists or it is empty then the user is prompted with a message ("File is empty or File does not exits).

Snap: [2] Read



```
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice: 2

*************[Contact-Book]*************
First name: Imran
Rol / Contact # 8978394734
Address: Ilamabad


--------------
First name: Qasim
Rol / Contact # 89778676765
Address: Karachi


--------------
```

Selecting the **Read** option all the existing data in contact book is displayed.



```
*************[Contact-Book]*************
First name: Imran
Rol / Contact # 8978394734
Address: Ilamabad

---------------
First name: Qasim
Rol / Contact # 89778676765
Address: Karachi

---------------
```

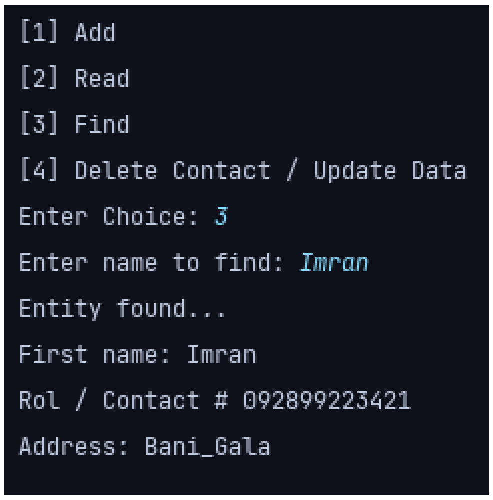
- (Feature) **Find Contact:**

This is a method of Contact Book derived from the method (**successful login**). A logged in (Registered user) can find data in his own contact book. This is done by reading the document and encounter the required contact details. After accessing this feature following possibilities can occur:

- Askes the user to enter the contact name to be found.
- If contact name matches, all the contact details would be displayed on terminal.
- If contact name doesn't matches then, the user is prompted with a message **("Contact not found")**

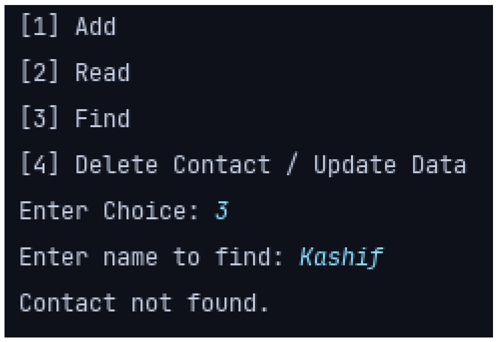
Snap:

Contact found



```
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice: 3
Enter name to find: Imran
Entity found...
First name: Imran
Rol / Contact # 092899223421
Address: Bani_Gala
```

Contact not found



```
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice: 3
Enter name to find: Kashif
Contact not found.
```

- (Feature) **Delete / Update Contact:**

This is a method of Contact Book derived from the method (successful login). This method initially askes the contact name to verify even if it exits or not. After logging in, if user selects the choice of **(Delete / Update Contact)** then he has got the **authority** to **delete** a specific contact or update that contact details. This method firstly, askes for two options:

- **Delete contact data.**
- **Update contact data.**

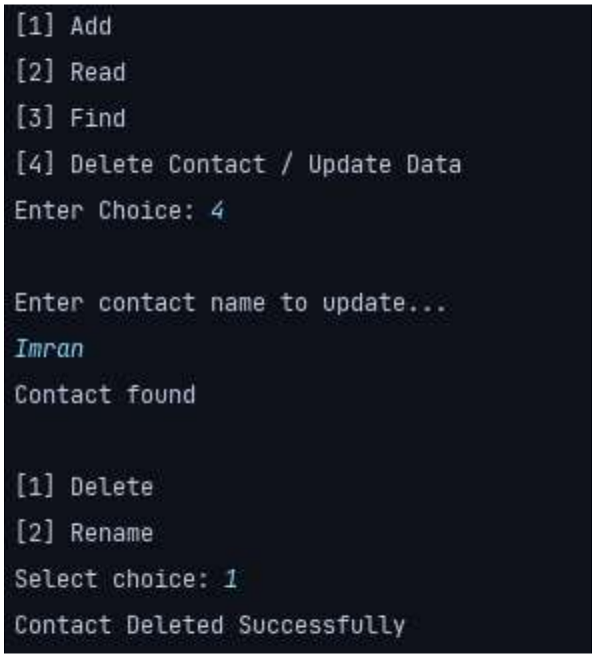
According to user choice he is redirected to the either method of deletion or updating, where they perform the tasks of deletion or updating.

- **Delete contact data:**

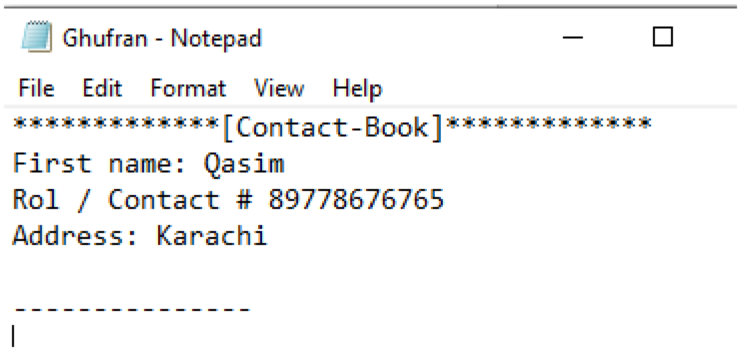
This is a method of Contact Book derived from the method (Delete/update method).

  - If the contact name is found then he has got the authority to delete. After deletion the user will prompted with message **("Contact successfully deleted").**
  - If contact name doesn't exists then the user is prompted with message **("Contact doesn't exists")**

Snap:



```
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice: 4

Enter contact name to update...
Imran
Contact found

[1] Delete
[2] Rename
Select choice: 1
Contact Deleted Successfully
```

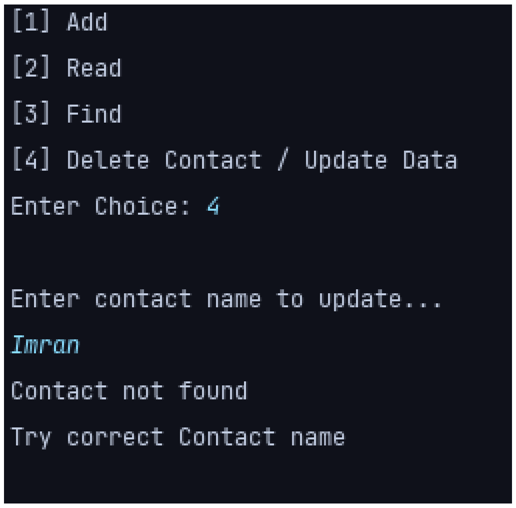


```
*************[Contact-Book]*************
First name: Qasim
Rol / Contact # 89778676765
Address: Karachi

---------------
```

In above figure the contact **Imran** has been deleted.

When the user selects the delete feature and after inserting the contact name, it will be permanently deleted from the disk and from contact book.

After Deletion:



```
[1] Add
[2] Read
[3] Find
[4] Delete Contact / Update Data
Enter Choice: 4

Enter contact name to update...
Imran
Contact not found
Try correct Contact name
```

After the contact has been **deleted successfully**. Now it doesn't exists and that is why it can't be read or found.

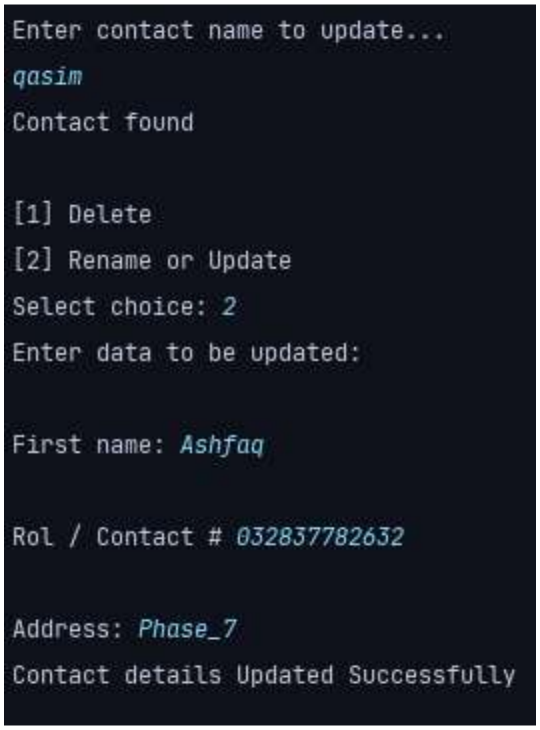
- **Update contact data:**

This is a method of Contact Book derived from the method (Delete/update method).

If the contact name is found then he has got the authority to Update. After updating data the user will prompted with message **("Contact updated deleted").**

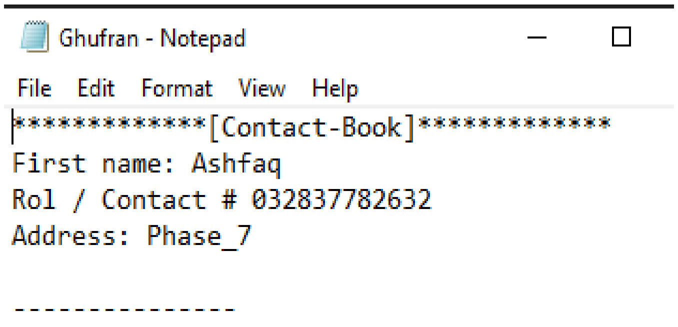
  - If contact name doesn't exists then the user is prompted with message **("Contact doesn't exists")**

Snap:



```
Enter contact name to update...
qasim
Contact found

[1] Delete
[2] Rename or Update
Select choice: 2
Enter data to be updated:

First name: Ashfaq

Rol / Contact # 032837782632

Address: Phase_7
Contact details Updated Successfully
```

The contact details of Qasim has been replaced with Ashfaq. The user will now see the updated details.



```
*************[Contact-Book]*************
First name: Ashfaq
Rol / Contact # 032837782632
Address: Phase_7

---------------
```

Class 2: Gamification – Arena:

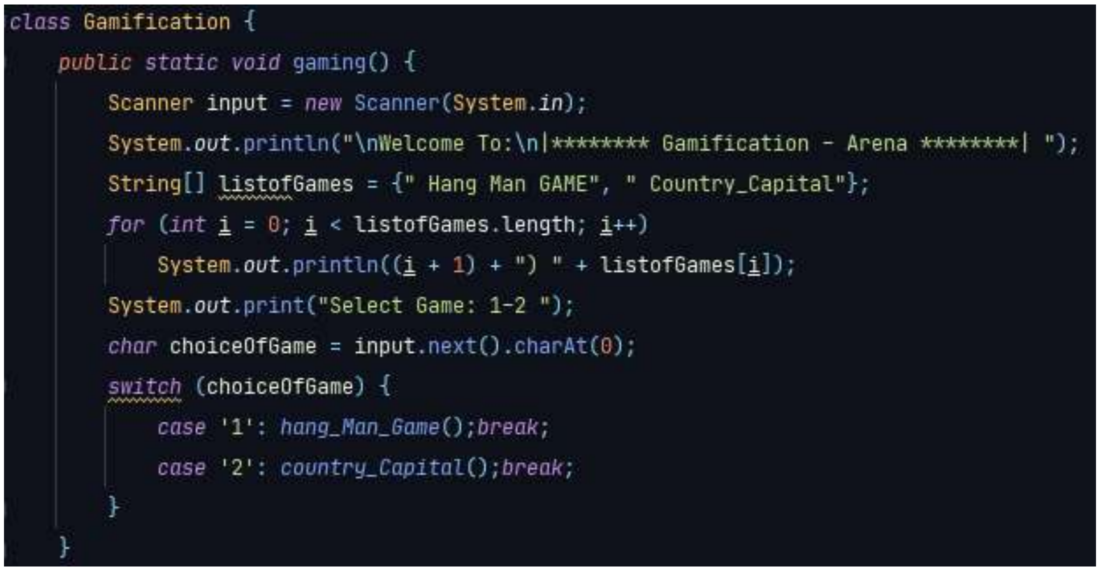
1. Gaming Method:

This method takes choice from the user by displaying menu.

Menu displays two games that the user can play.

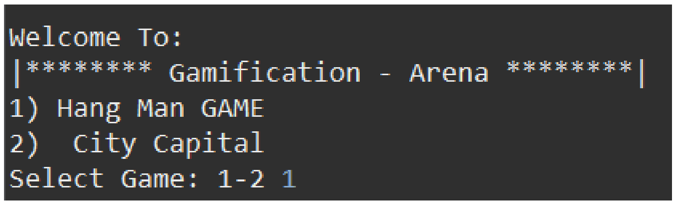
1) Hangman Game
2) Country Capital

- A **for loop** is used to display the options.
- A **switch** statement is used to control user input.
- Pressing 1 **invokes** the method hang_Man_Game()
- Pressing 2 **invokes** the method CountryCapital()
- If the user presses invalid key, then **try catch block** will catch the error and asks again to enter a proper input.
- **Do while** loop is user throughout to continuously ask for next **iteration**.



```
class Gamification {
    public static void gaming() {
        Scanner input = new Scanner(System.in);
        System.out.println("\nWelcome To:\n|******** Gamification - Arena ********| ");
        String[] listofGames = {" Hang Man GAME", " Country_Capital"};
        for (int i = 0; i < listofGames.length; i++)
            System.out.println((i + 1) + ") " + listofGames[i]);
        System.out.print("Select Game: 1-2 ");
        char choiceOfGame = input.next().charAt(0);
        switch (choiceOfGame) {
            case '1': hang_Man_Game();break;
            case '2': country_Capital();break;
        }
    }
```

Snap: (Getting user choice)



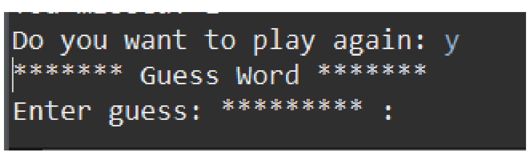
```
Welcome To:
|******** Gamification - Arena ********|
1) Hang Man GAME
2)  City Capital
Select Game: 1-2 1
```

1.1 Game 1: Hangman

- **Hangman Game Module:**

  - This module calls the random word module to take a random list of words.
  - Then it calls the play game module to start playing game.
  - At the end, it ask from user to play it again or not. If user enters "y", the while loop executes again and if user enters "n", the while loop breaks.

Snap:



```
Do you want to play again: y
******* Guess Word *******
Enter guess: ********* :
```

1.2 Random Word Module:

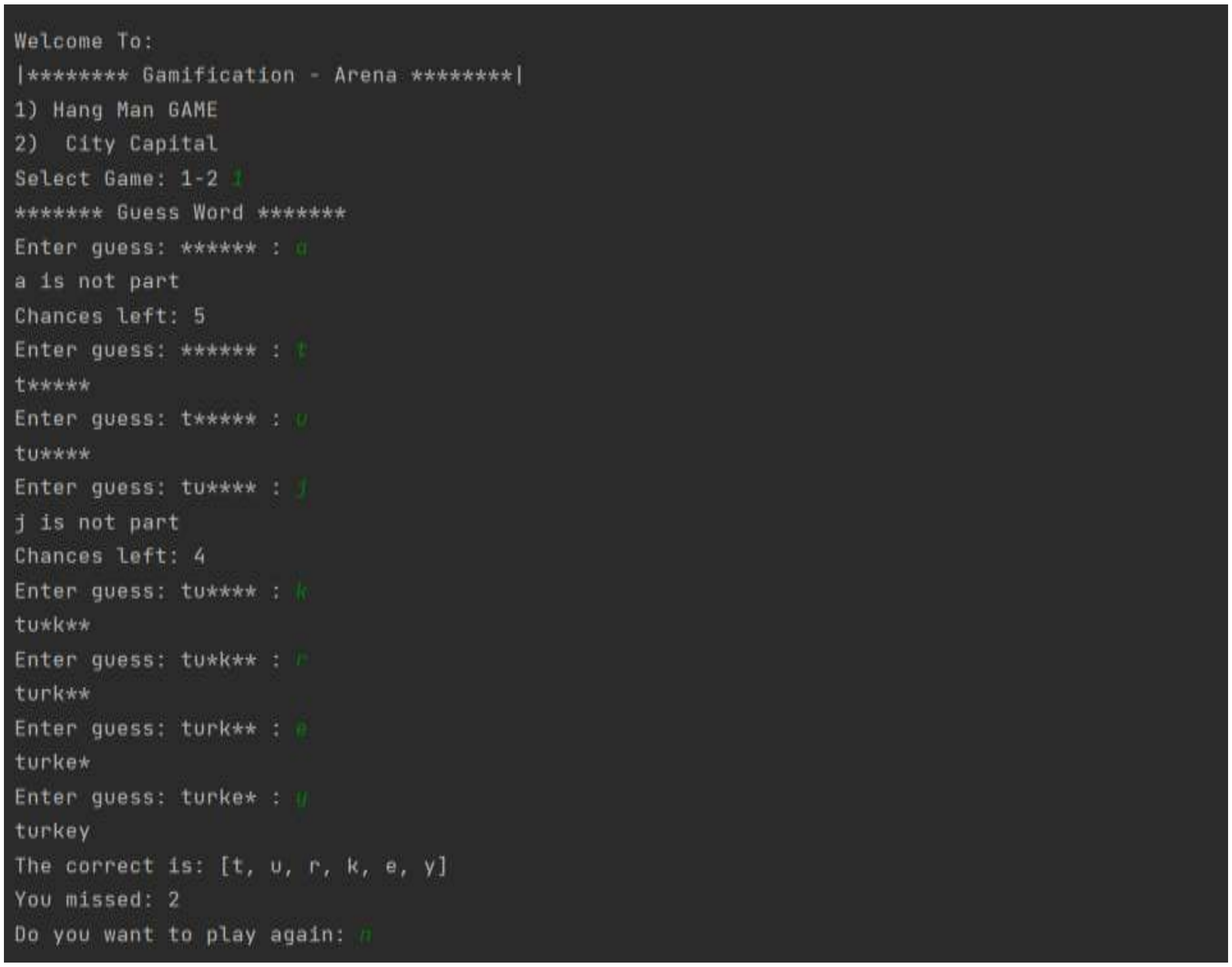
- This method contains four character array of 4 words and return one of the random char array to the **hangman game** module by using **Math.random()** module.

- **Play Game Module:**

This method takes a character list as a parameter that was being returned by the **random word** module.

1) This module ask from user to guess a letter in a word one by one by giving 5 chances.
2) If the letter is not present in the word, it tells that this letter is not the part of word and decrement the chance by one.
3) If the user guesses the correct letter, it simply replaces the asterisk with the letter.
4) When user loses all of the 5 chances, the program displays the end results.
5) At the end, it displays the correct word and the number of chances user has lost.

Snap: (The complete output)



```
Welcome To:
|********* Gamification - Arena *********|
1) Hang Man GAME
2)  City Capital
Select Game: 1-2 1
******* Guess Word *******
Enter guess: ****** : a
a is not part
Chances left: 5
Enter guess: ****** : t
t*****
Enter guess: t***** : u
tu****
Enter guess: tu**** : j
j is not part
Chances left: 4
Enter guess: tu**** : k
tu*k**
Enter guess: tu*k** : r
turk**
Enter guess: turk** : e
turke*
Enter guess: turke* : y
turkey
The correct is: [t, u, r, k, e, y]
You missed: 2
Do you want to play again: n
```

2. Game 2: Country Capital Game

- ### Country Capital Module:

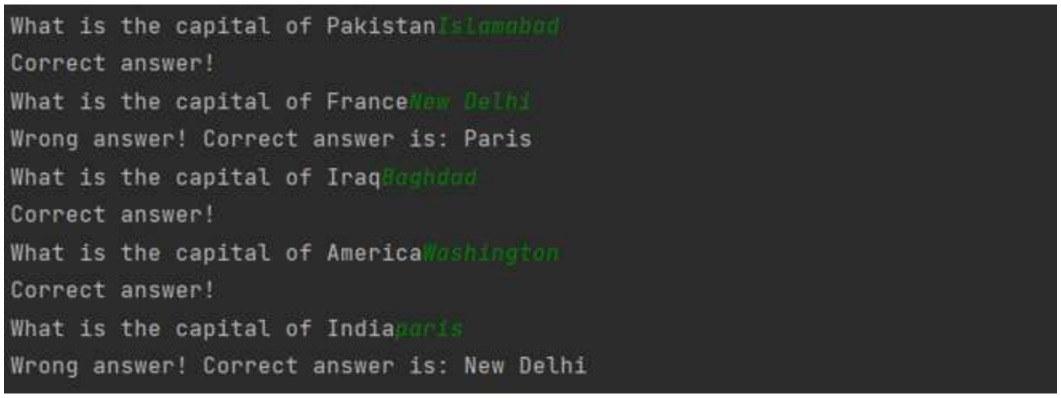
1) This module calls the shuffle module.

- ### Shuffle module:

It shuffles the rows in **2D array** that contains questions and their answers, so that random questions will be asked from user.

2) Coming back to the Country Capital module, it asks five random questions from user regarding a country's capital.

Snap: (Asking Questions)

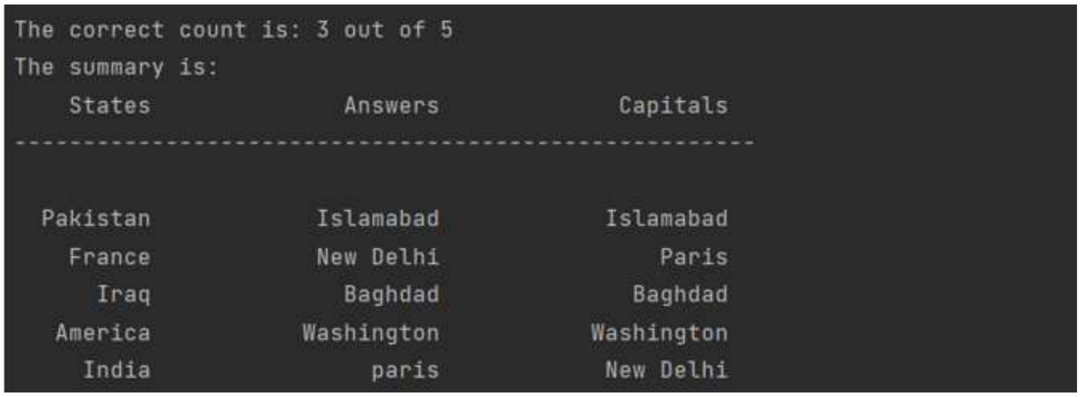


```
What is the capital of PakistanIslamabad
Correct answer!
What is the capital of FranceNew Delhi
Wrong answer! Correct answer is: Paris
What is the capital of IraqBaghdad
Correct answer!
What is the capital of AmericaWashington
Correct answer!
What is the capital of Indiaparis
Wrong answer! Correct answer is: New Delhi
```

3) It uses a **2D array** to store summary having 5 rows to store 5 results and 3 column to store question, user's answer and correct answer.

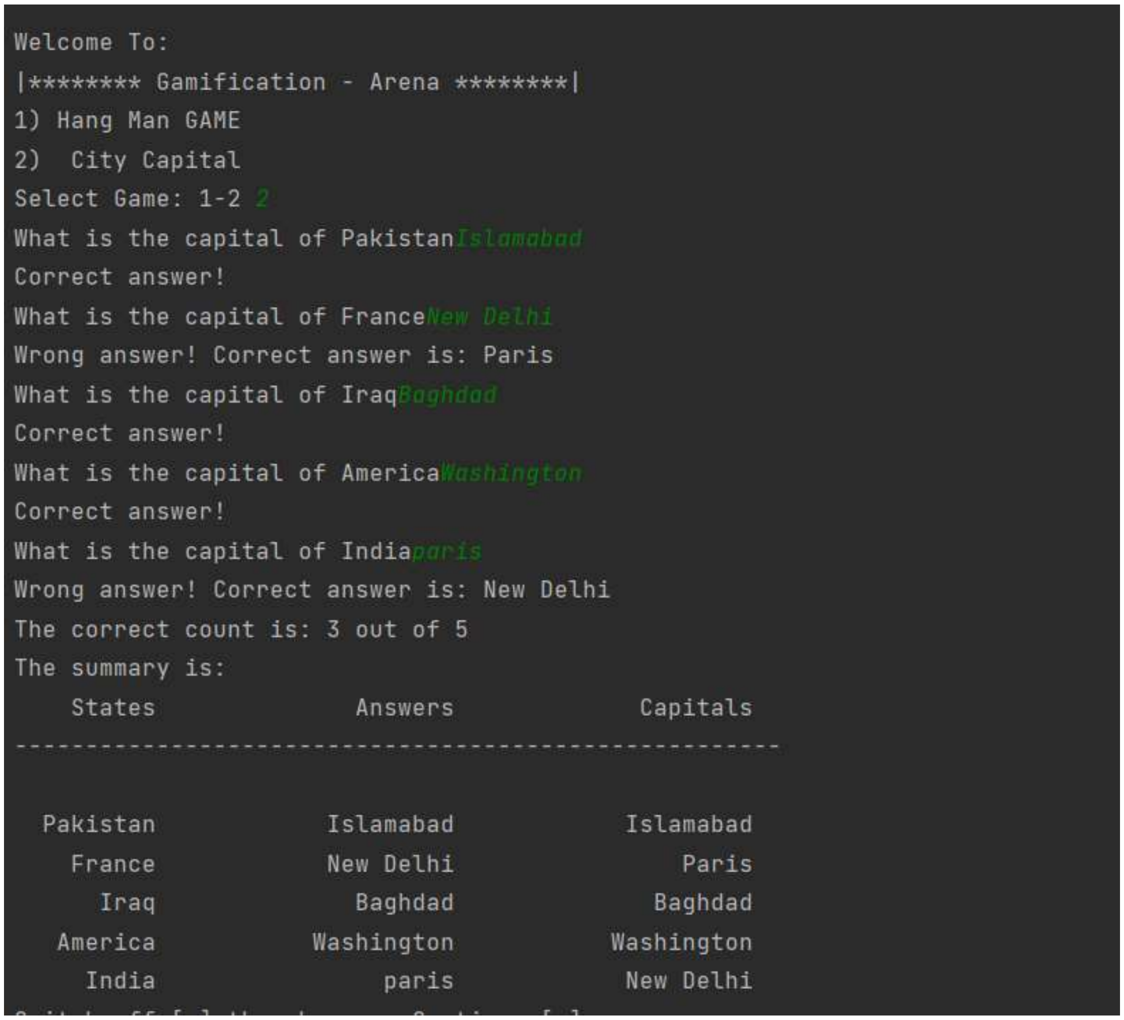
4) In the end, it displays correct count of user along with complete summary of the game.

Snap: (Displaying Summary)



```
The correct count is: 3 out of 5
The summary is:
    States              Answers             Capitals
------------------------------------------------------


  Pakistan            Islamabad            Islamabad
    France            New Delhi                Paris
      Iraq              Baghdad              Baghdad
   America           Washington           Washington
     India                paris            New Delhi
```

Snap: (Complete output)



```
Welcome To:
|******** Gamification - Arena ********|
1) Hang Man GAME
2)  City Capital
Select Game: 1-2 2
What is the capital of PakistanIslamabad
Correct answer!
What is the capital of FranceNew Delhi
Wrong answer! Correct answer is: Paris
What is the capital of IraqBaghdad
Correct answer!
What is the capital of AmericaWashington
Correct answer!
What is the capital of Indiaparis
Wrong answer! Correct answer is: New Delhi
The correct count is: 3 out of 5
The summary is:
    States              Answers              Capitals
------------------------------------------------------


  Pakistan            Islamabad             Islamabad
    France            New Delhi                 Paris
      Iraq              Baghdad               Baghdad
   America           Washington            Washington
     India                paris             New Delhi
```

Smooth transfer of Control, Exception Handling, File Handling:

Throughout the project we used:

- A **for loop** is used to display the options.
- A **switch** statement is used to control user input.
- Pressing a user choice invokes the method accordingly.
- If the user presses invalid key, then **try catch block** will catch the error and asks again to enter a proper input.
- **Do while** loop is user throughout to continuously ask for next **iteration**.
- For File handling and continuous storing and reading of data, we use file objects like:
    - Scanner object. (For reading)
    - Filewriter object.(For writing).
    - File.delete() method for deletion.