An internship report on **JAVA DEVELOPMENT**

A report submitted in partial fulfilment of the requirements for the curriculum of 6th Semester, Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

PRASHANT KUMAR

Regd. No: 20033445017

(Duration- 2nd May 2023 - 10th June 2023)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAMGARH ENGINEERING COLLEGE

(ESTD. By Govt. Of Jharkhand & Run by Techno India)

Approved by AICTE, New Delhi and affiliated to JUT, Ranchi

ISO 9001:2015, Certified Institute

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAMGARH ENGINEERING COLLEGE

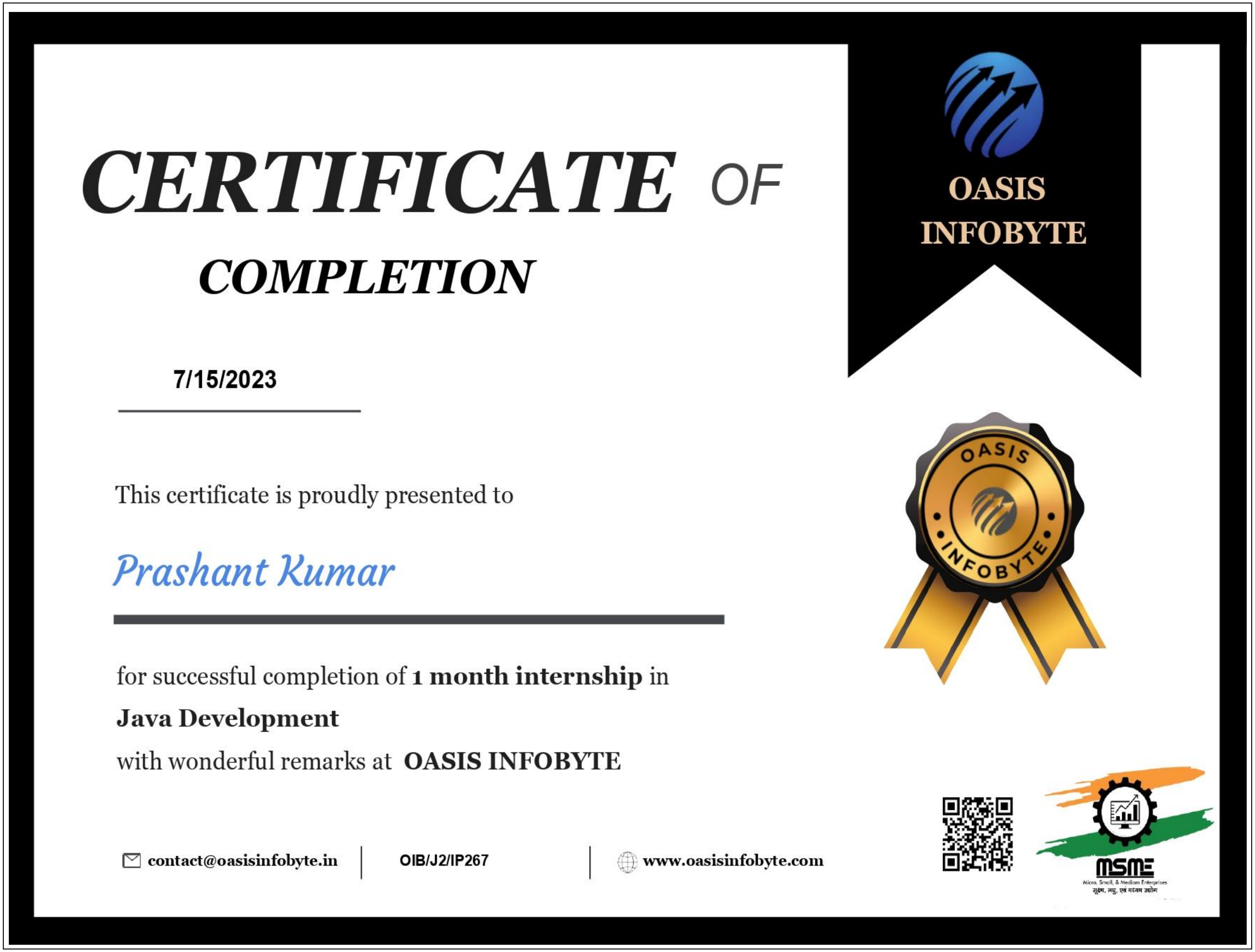
(ESTD. By Govt. Of Jharkhand & Run by Techno India)

This is to certify that the one month virtual **"Internship report"** submitted by **PRASHANT KUMAR** (Regd. No. 20033445017) in the domain **"JAVA DEVELOPMENT"** (2nd May to 10th June) is work done by him and submitted during 2023-2024 academic year, in partial fulfillment of the requirements for the curriculum of 6th Semester, degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING,** at **OASIS INFOBYE.**

Prof. Ashim Kumar Mahto

Head Of the Department

Department Of CSE



CERTIFICATE OF COMPLETION

OASIS INFOBYTE

7/15/2023

This certificate is proudly presented to

Prashant Kumar

for successful completion of **1 month internship** in **Java Development** with wonderful remarks at **OASIS INFOBYTE**

contact@oasisinfobyte.in | OIB/J2/IP267 | www.oasisinfobyte.com

MSME
Micro, Small, & Medium Enterprises
सूक्ष्म, लघु, एवं मध्यम उद्यम

Table of Contents

Acknowledgements

I would like to express our deepest appreciation to all those who provided me the possibility to complete this Internship. A special gratitude to the **Team of OASIS INFOBYTE** for giving me the opportunity to do an internship within the organization. Your patience and understanding allowed me to grow and learn from my experiences, making this internship an enriching and rewarding one.

A special thanks goes to the Head of the Department, **Prof. Ashim Kumar Mahto** whose suggestions and guidance helped me and specially encouraged me for making this report, I also appreciate the guidance and help received from all other teachers.

I am extremely grateful to my friends and the fellow interns who helped me in successful completion of this internship.

Prashant Kumar

(20033445017)

Internship Objectives

- Internships are generally thought of to be reserved for college students looking to gain experience in a particular field. However, a wide array of people can benefit from Training Internships in order to receive real world experience and develop their skills.

- An objective for this position should emphasize the skills you already possess in the area and your interest in learning more.

- Internships are utilized in a number of different career fields, including architecture, engineering, healthcare, economics, advertising and many more.

- Some internship is used to allow individuals to perform scientific research while others are specifically designed to allow people to gain first-hand experience working.

- Utilizing internships is a great way to build your resume and develop skills that can be emphasized in your resume for future jobs. When you are applying for a Training Internship, make sure to highlight any special skills or talents that can make you stand apart from the rest of the applicants so that you have an improved chance of landing the position.

- This internship is a task-based internship the so, focus is on accomplishing specific objectives or assignments and need not follow the traditional approach.

Introduction

Java is a versatile, high-level, object-oriented programming language that has gained immense popularity since its inception in 1995. Developed by James Gosling and his team at Sun Microsystems (now owned by Oracle Corporation), Java was designed to be platform-independent, enabling developers to write code that could run on any device or operating system. Over the years, Java has become a cornerstone of the software development industry, powering a wide range of applications from web and mobile to enterprise-level systems.

Key Features of Java:

Platform Independence: Java's "Write Once, Run Anywhere" (WORA) principle allows developers to create applications that can be executed on different platforms without modification, thanks to the Java Virtual Machine (JVM) that acts as an intermediary between the code and the underlying hardware.

Object-Oriented Programming (OOP): Java follows the OOP paradigm, enabling developers to model real-world entities using classes and objects. This approach promotes code reusability, maintainability, and scalability.

Garbage Collection: Java features an automatic garbage collector that manages memory allocation and frees up resources when objects are no longer in use. This reduces the burden on developers for memory management.

Exception Handling: Java provides a robust mechanism for handling errors and exceptions, making code more robust and reliable.

Rich Standard Library: Java comes with an extensive standard library (Java API), offering a wide range of pre-built classes and methods for various tasks, such as networking, file I/O, and data structures.

Applications of Java:

Web Development: Java has been widely used for creating web applications through technologies like JavaServer Pages (JSP), Servlets, and the Spring Framework.

Android App Development: Java has been a primary programming language for developing Android mobile applications.

Enterprise Applications: Java's scalability and reliability make it a popular choice for building large-scale enterprise applications and systems.

Internet of Things (IoT): Java's lightweight version, Java ME (Micro Edition), is used in developing applications for IoT devices.

Big Data and Analytics: Java is employed in various big data processing frameworks like Apache Hadoop and Apache Spark.

Tasks

As I have enrolled for a task-based internship which typically refers to an internship program where interns are assigned specific tasks or projects to complete during their internship period. Instead of following a traditional time-based approach where interns work for a fixed number of hours each day, the focus is on accomplishing specific objectives or assignments.

In a task-based internship, interns may work on real projects that align with the company's goals and provide meaningful learning experiences. The tasks given to interns are usually relevant to their field of study or career interests, allowing them to gain practical skills and hands-on experience.

Here are some characteristics of task-based internships:

- **Goal-Oriented:** The internship revolves around achieving specific goals or completing designated tasks rather than merely fulfilling a set number of hours.
- **Project-Centered:** Interns may be assigned to work on individual projects or collaborate with teams on ongoing initiatives.
- **Skill Development:** The tasks assigned aim to help interns develop specific skills related to their chosen career path.
- **Practical Experience:** Interns get the opportunity to apply theoretical knowledge to real-world scenarios, enhancing their understanding and expertise.

- **Evaluation:** Interns may be assessed based on their performance and the quality of work completed during the internship.
- **Flexibility:** Task-based internships may offer more flexibility in terms of work hours, as long as the assigned tasks are completed within the specified timeframe.

Tasks completed by me:

To fulfill the eligibility criteria, I have completed some of the tasks using Java programming:

1. Number Guessing Game
2. Atm Interface

A. Number Guessing Game:

This Java program implements a number guessing game. The game randomly generates a number between 1 and 100 and prompts the user to guess the number within 5 trials. The program provides feedback for each guess, indicating whether the actual number is greater or less than the user's guess. If the user exhausts all 5 trials without guessing the correct number, the program reveals the actual number.

(i) Code:

```
// Java program to guess a number randomly generated by this program.

package Task1;

import java.util.Scanner;

public class NoGuessingGame {

    // Function that implements the number guessing game.
    public static void NumberGuessingGame()
    {
```

```
Scanner sc = new Scanner(System.in);
int number = 1 + (int)(100* Math.random());//type casting

int T = 5;//t=trails
int i, guess;

System.out.println("A number is chosen between 1 to 100. \n" +
        "Guess the number within 5 trials.");

// Iterate over 5 Trials
for (i = 0; i < T; i++) {

    System.out.println(
            "Guess the number:");

    // Take input for guessing the number
    guess = sc.nextInt();

    if (number == guess) {
        System.out.println(
                "Congratulations! You guessed the number in  +T );
        break;
    }
    else if (number > guess && i != T - 1) {
        System.out.println(
                "The number is greater than " + guess);
    }
    else if (number < guess && i != T - 1) {
        System.out.println(
                "The number is less than " + guess);
    }
}

if (i == T) {
```

```
            System.out.println(
                "Sorry! You have exhausted your trials ;( \n" +
                    "Play again ;) ");

            System.out.println(
                "The number was " + number);
        }
    }

    public static void main(String[] arg)
    {
        // Function Call
        NumberGuessingGame();
    }
}
```

(ii) Code Walkthrough:

1. The program starts by importing the **'Scanner'** class to read input from the user.

2. The **'game'** class is defined, and the **'NumberGuessingGame'** function is declared as a static method inside the class.
3. Inside the **'NumberGuessingGame'** function:
   - A random number between 1 and 100 is generated and stored in the variable **'number'**.
   - The maximum number of trials (**'T'**) is set to 5, and variables **'i'** and **'guess'** are declared.
   - The user is informed about the game rules through a print statement.
   - The program enters a **'for'** loop to allow the user to make up to 5 guesses:
   - The user is prompted to enter their guess through a print statement.
   - The user's input is read using **'sc.nextInt()'** and stored in the variable **'guess'**.
   - If the user's guess matches the randomly generated number (number), the program prints a congratulatory message and breaks out of the loop.

- If the user's guess is incorrect and not the last trial, the program provides a hint indicating whether the actual number is greater or less than the user's guess.

4. If the user exhausts all 5 trials without guessing the correct number (i.e., **i ==T**), the program prints a message informing the user that they have run out of trials and reveals the actual number.
5. The **'main'** function is called, which in turn calls **'NumberGuessingGame'** function to start the game.

(iii)<u>Conclusion</u>:

The number guessing game implemented in the provided Java code is a simple and interactive program. It allows users to test their guessing skills by attempting to guess a randomly generated number within a limited number of trials. The program provides helpful hints to guide the user in finding the correct number and informs them when they have either guessed correctly or exhausted their trials.

The program can be further improved by adding more features, such as validating user input to ensure they enter a valid number within the specified range. Additionally, a scoring system or leaderboard could be introduced to track and display the user's performance across multiple game sessions. Overall, the current implementation serves as an enjoyable and engaging game for users to play and practice their guessing abilities.

B. <u>Atm Interface</u>:

The provided code is a simple command-line Automated Teller Machine (ATM) example in Java. It simulates a basic banking application where users can perform various operations such as withdrawing money, depositing money, checking their balance, transferring money to other accounts, and viewing transaction history.

(i) <u>Code</u>:

```
//import required classes and packages
package Task2;
import java.util.Scanner;
```

```
public class ATMExample
{
    //main method starts
    public static void main(String args[])
    {
        float balance = 55000F;
        float amount;
        int transactions = 0;
     String transactionHistory = "";
        String name;

        //create scanner class object to get choice of user
        Scanner sc = new Scanner(System.in);
        while(true)  {
            System.out.println("Automated Teller Machine\n"+
                    "Choose 1 for Withdraw\n"+
                    "Choose 2 for Deposit\n" +
                    "Choose 3 for Check Balance\n"+
                    "Choose 4 for Transfer\n"
                    +"Choose 5 for Transaction History\n"+
                    "Choose 6 for EXIT" );

            //get choice from user
            int choice = sc.nextInt();
            switch(choice)
            {
case 1:
        System.out.print("Enter money to be withdrawn:");
//get the amount to withdraw from user
        amount = sc.nextFloat();
        if(balance >= amount )
        {
            transactions++;
            balance = balance - amount;
            System.out.println("Please collect your money.");
            String str = amount + " Rs Withdrawn.\n";
            transactionHistory = transactionHistory.concat(str);
        }
        else
        {
    //show error message is the amount is no available
            System.out.println("Insufficient Balance.");
        }
        System.out.println("");
        break;
     case 2:
        System.out.print("Enter money to be deposited:");

        //get the amount to deposit from the user
        amount = sc.nextFloat();
        transactions++;
        //add the deposit amount to the total balance
        balance = balance + amount;
        System.out.println("Your Money has been successfully deposited.");
        System.out.println("");
        String str = amount + " Rs deposited.\n";
        transactionHistory = transactionHistory.concat(str);
        break;
    case 3:
        System.out.println("Balance : "+ balance);
        System.out.println("");
              break;
    case 4:
        //Transfer money to other accounts
        System.out.println("\nEnter Receipent's Name : ");
        name= sc.next();
        System.out.println("\nEnter Receipent's Account_Number : ");
        int acc = sc.nextInt();
```

```
            System.out.println("\nEnter amount : ");
            amount = sc.nextFloat();
            if(balance >= amount)
      {
        transactions++;
        //remove the withdrawl amount from the total balance
        balance = balance - amount;
        System.out.println("Transfer Successful.");
        String str1 = amount + " Rs transfered to " +name +" with account no " + acc +
        ".\n";
        transactionHistory = transactionHistory.concat(str1);
        }
        else
            {
              //show error message
              System.out.println("Insufficient Balance.");
            }
        System.out.println("");
        break;
        case 5:
             if ( transactions == 0 ) {
             System.out.println("\nEmpty");}
        else {
              System.out.println("\n" + transactionHistory);}
                System.out.println("");
                break;
                  case 6:
            //exit from the menu
            System.exit(0);
             } } } }
```

(ii) <u>Code Walkthrough</u>:

1. The program starts by declaring and initializing several variables, including '**balance**' (initial balance of 55000), **'amount'** (used to store the amount to be withdrawn or deposited), **'transactions'** (to keep track of the total number of

transactions), and **‘transactionHistory’** (a string to store the history of all transactions).

2. The **‘Scanner’** class is used to take user input from the console.
3. The program enters a perpetual loop (**‘while(true)’**) that presents the user with a menu of available options.
4. The user is prompted to choose an operation from the menu using an integer input (choice) obtained through **‘sc.nextInt()’**.
5. The user's input is then matched using a **‘switch-case’** statement to perform the corresponding operation based on their choice.

Supported Operations:

1.Withdraw (Choice 1):

- The user is asked to enter the amount to be withdrawn (**‘amount’**).
- If the available **‘balance’** is greater than or equal to the requested amount, the withdrawal is processed.
- The **‘balance’** is updated, and a success message is displayed.
- The transaction details are appended to the **‘transactionHistory’** string.

2.Deposit (Choice 2):

- The user is asked to enter the amount to be deposited (**‘amount’**).
- The deposited amount is added to the **‘balance’**.
- A success message is displayed, and the transaction details are appended to the **‘transactionHistory’** string.

3.Check Balance (Choice 3):

- The current **‘balance’** is displayed.

4.Transfer (Choice 4):

- The user is asked to enter the recipient's name (**‘name’**), account number (**‘acc’**), and the amount to be transferred (**‘amount’**).
- If the available **‘balance’** is greater than or equal to the requested amount, the transfer is processed.
- The recipient's account **‘balance’** is updated (**‘balance** = **balance** – **amount**’), and the sender's **‘balance’** is deducted by the transferred amount.
- A success message is displayed, and the transaction details are appended to the **‘transactionHistory’** string.

5. Transaction History (Choice 5):

If there are no transactions (**‘transactions == 0’**), an "Empty" message is displayed.

Otherwise, the **‘transactionHistory’** containing all transaction details is shown.

6.Exit (Choice 6):

The program terminates when the user chooses this option.

(iii) <u>Conclusion</u>:

The provided code demonstrates a basic ATM-like banking application using Java, allowing users to perform various banking operations. It employs a simple loop and switch-case structure to handle different user choices and maintains a transaction history for reference. Note that this code is meant for educational purposes and lacks features that would be present in a real-world ATM system, such as user authentication, database integration, and error handling for edge cases.

<u>About Internship</u>

Some important points we have to remember during this internship tenure –

- Update your LinkedIn Profile and share all your achievements (Offer Letter / Internship Completion Certificate) which you got from us and tag Oasis Infobyte, also use #oasisinfobyte.
- If your project/code is found copied, your internship will be terminated and you will be banned from further opportunities from us.
- Share video of the completed task on LinkedIn and tag Oasis Infobyte also use #oasisinfobyte (Explanation is Mandatory).
- For Tech Internship maintain a separate GitHub repository (name as OIBSIP for all the tasks and share the link of the GitHub repository in the task submission form (it will be given later through email).

I successfully fulfilled each criteria and finally awarded with the Internship certificate.

One can easily apply for this virtual internship by visiting the official website "**https://oasisinfobyte.com/#**". It is an unpaid and task-based internship which can be easily done from anywhere remotely.

Conclusion

This internship covers essential aspects of Java programming, task-based internships, and demonstrates two practical Java programs.

The first Java program, the Number Guessing Game, is a simple interactive game where the user attempts to guess a randomly generated number within a limited number of trials. The program provides helpful hints and informs the user when they guess correctly or exhaust all trials.

The second program, the ATM Interface, simulates a basic banking application. Users can perform operations like withdrawing, depositing, checking balance, transferring money, and viewing transaction history. The program maintains the user's balance, transaction history, and provides a user-friendly menu for selecting operations.

Both programs demonstrate fundamental programming concepts, including conditional statements, loops, functions, and user input/output handling. While these implementations serve as educational examples, they can be further enhanced to include error handling, input validation, and additional features for real-world applications.

For interns, task-based internships offer a goal-oriented and skill-focused learning experience. Working on specific projects allows them to apply theoretical knowledge to practical scenarios, develop relevant skills, and gain hands-on experience in their chosen field. The internship's flexibility and project-centered approach provide valuable learning opportunities, helping interns build a strong foundation for their future careers.