INDUSTRIAL TRAINING INTERNSHIP REPORT

INDUSTRIAL TRAINING SEMINAR REPORT TOPIC

JAVA PROGRAMMING INTERNSHIP

Submitted in partial fulfillment of the degree of

B tech

Gandhi Institute of Excellent Technocrats (GIET)

BY

Md Ezaz

24LE25

Gandhi Institute of Excellent Technocrats (GIET)

Ghangapatna , bhubaneswar, Odisha

CERTIFICATE

This is to certify that Industrial Training Internship Report entitled "**JAVA PROGRAMMING**" has been submitted by 24LE25 **Md Ezaz** for partial fulfillment of the degree of B tech from Gandhi Institute of Excellent Technocrats (GIET) .It is found satisfactory and approved for submission.

Date:

10/10/2024

Bibhuti bhusan behera

HOD

Gandhi Institute of Excellent Technocrats (GIET)

Subhrajit pradhan
Principal / Director

Gandhi Institute of Excellent Technocrats (GIET)

ATTENDANCE CERTIFICATE

ThIs is to certify that Md Ezaz has successfully completed an internship programme at INTERNPE

During this period, [he/she] demonstrated commendable dedication and enthusiasm towards [his/her] assigned tasks and responsibilities. [He/She] actively participated in various projects and initiatives, contributing positively to the team's objectives.

Internship Supervisor: **Krati Kumari**

This certificate is awarded in recognition of [his/her] commitment and valuable contribution to **INTERNPE**.

Authorized Signature: 

HR

INTERNPE

InternPE
Jaipur Rajasthan

OFFER LETTER

IPI#41708

Dear MD EZAZ,

We are delighted to welcome you for the Internship Program of **JAVA Programming** being observed by INTERNPE, as a learning opportunity for you.

Your Internship starts on **01/10/2024** and ends on **31/10/2024**. It's **04 Weeks** program. It's also an unpaid program all focus is on learning.

We look forward to a worthwhile and faithful association that will make you equipped for future projects. Wishing you the most enjoyable and truly meaningful internship program experience.

(Co-Founder)
InternPe

https://internpe.in/

INTERNSHIP MENTOR DECLARATION

This is to certify that the Industrial Training Internship In **INTERNPE** entitled **JAVA PROGRAMMING INTERNSHIP** by Md Ezaz has been done successfully and completed all the tasks provided in the internship.

Date:
10/10/2024

Ms. KRATI PORWAL

Mentor
www.interpe.in

TASK DESCRIPTION

TASK 1:

Guess a Number Task:

OBJECTIVE:

1. Random Number Generation: Generate a random number between a specified range.
2. User Input: Allow the user to guess the generated number.
3. Comparison: Compare the user's guess with the generated number.
4. Feedback: Provide feedback indicating whether the guess is correct, too high, or too low.

HOW TO PERFORM:

1. Random Number Generation:
   Use Java's `Random` class or another method to generate a random number within a specified range.

2. User Input:
   Implement a mechanism to allow the user to input their guessed number.

3. Comparison:
   Compare the user's guessed number with the randomly generated number to determine if it's correct, too high, or too low.

4. Feedback:
   Provide clear feedback to the user based on the comparison, indicating whether the guess is correct or providing a hint for the next attempt.

5. Iteration (Optional):
   Optionally, implement a loop to allow the user multiple attempts until they correctly guess the number or choose to exit the game.

6. User Interaction:
   Ensure a user-friendly experience with clear instructions, and possibly add features like prompting the user to play again.

7. Scalability (Optional):
   If desired, consider making the game scalable by allowing users to choose the range of numbers or introducing difficulty levels.

8. Testing:
   Test the game thoroughly to ensure that it works as expected, including scenarios where the user guesses correctly or incorrecty

9. Documentation:
   Provide comments and documentation to explain the functionality of the code for future reference or collaboration.

TASK 2
ROCK PAPER SESSIOR

1. User Input: Allow players to input their choice of rock, paper, or scissors.
2. Random Selection: Generate a random choice for the computer opponent.
3. Comparison: Compare user and computer choices to determine the winner based on the Rock-Paper-Scissors rules.
4. Feedback: Provide clear feedback indicating the winner or if it's a tie.

HOW TO PERFORM:

1. User Input:
   - Allow players to input their choice of rock, paper, or scissors using user interaction mechanisms.

2. Random Selection:
   - Generate a random choice for the computer opponent. Utilize Java's `Random` class or another method for randomness.

3. Comparison:
   - Implement logic to compare the user's choice with the computer's choice based on the Rock-Paper-Scissors rules to determine the winner.

4. Feedback:
   - Provide clear feedback to the user, indicating the winner or declaring a tie. This could be done through console output or a graphical user interface.

5. User Interaction:
   - Ensure a user-friendly experience with clear instructions, and consider providing options for the user to play again or exit the game.

6. Scalability (Optional):
   - Optionally, consider making the game scalable by adding more choices or introducing additional features, such as a scoring system.

7. Testing:
   - Test the game thoroughly to ensure it works as expected under various scenarios, including winning, losing, and tying.

8. Documentation:
   - Provide comments and documentation to explain the functionality of the code for future reference or collaboration.

TASK 3:

TIC TAC TOE TASK:

OBJECTIVE:

1. e a 3x3 grid to represent the Tic Tac Toe game board.
2. Player Turns: Allow two players to take turns marking spaces with 'X' and 'O'.
3. Winning Conditions: Define conditions for a player to win by having three marks in a row, column, or diagonal.
4. User Experience: Ensure a user-friendly experience with clear instructions, valid move validation, and an option to replay.

HOW TO PERFORM:

1. Grid Setup:
   - Design a 3x3 grid to represent the Tic Tac Toe game board.

2. Player Turns:
   - Allow two players to take turns marking spaces with 'X' and 'O'.
   - Implement a mechanism to switch between players after each move.

3. User Input:
   - Enable user interaction by allowing players to input their moves on the grid.

4. Winning Conditions:
   - Define winning conditions by checking for three marks in a row, column, or diagonal.
   - Implement logic to detect and announce a winner.

5. Tie Detection:
   - Implement logic to detect a tie when all spaces on the grid are filled and no player has won.

6. Feedback:
   - Provide clear feedback after each move, indicating the current player, the outcome of the move, and game status.

7. Reset and Replay:
   - Include functionality to reset the game board for a new match.
   - Allow players to replay without needing to restart the program.

8. Validation:
   - Validate moves to ensure players can only mark empty spaces and avoid illegal moves.

9. User Interface Enhancements:
   - Consider adding user-friendly features such as highlighting winning combinations or animating the mark placements.

10. Testing and Debugging:
   - Test the game thoroughly to ensure smooth gameplay, including edge cases like invalid inputs, and debug any issues.

11. Optional Features (if time allows):
   - Explore additional features like a scoreboard, customizable player marks, or a graphical user interface.

TASK 4:

CONNECT 4 GAME TASK:

OBJECTIVE:

1. Grid Structure: Create a 6x7 grid to represent the Connect 4 game board.
2. Player Interaction: Allow two players to take turns dropping colored discs into columns.
3. Winning Conditions: Define conditions for a player to win by connecting four discs horizontally, vertically, or diagonally.
4. User Experience: Ensure a user-friendly experience with clear instructions, valid move validation, and an option to replay.

HOW TO PERFORM:

Certainly! Here's a non-code, step-by-step guide on how to create a Connect 4 game in Java:

1. Grid Setup:
   - Design a 6x7 grid to represent the Connect 4 game board.

2. Player Turns:
   - Allow two players to take turns dropping colored discs into columns.
   - Implement a mechanism to switch between players after each move.

3. Disc Placement:
   - Enable user interaction by allowing players to drop their discs into columns.
   - Implement logic to place the disc in the lowest available position within the selected column.

4. Winning Conditions:
   - Define winning conditions by checking for four consecutive discs in a row, column, or diagonal.
   - Implement logic to detect and announce a winner.

5. Tie Detection:
   - Implement logic to detect a tie when the grid is full with no Connect 4.

6. Feedback:
   - Provide clear feedback after each move, indicating the current player, the outcome of the move, and game status.

7. Reset and Replay:
   - Include functionality to reset the game board for a new match.
   - Allow players to replay without needing to restart the program.

8. Validation:
   - Validate moves to ensure players can only drop discs in valid columns and avoid illegal moves.

9. User Interface Enhancements:
   - Consider adding user-friendly features such as highlighting winning combinations or animating the disc placements.

10. Testing and Debugging:
   - Test the game thoroughly to ensure smooth gameplay, including edge cases like invalid inputs, and debug any issues.

11. Optional Features (if time allows):
   - Explore additional features like a scoreboard, customizable disc colors, or a graphical user interface.

CONCLUSION

My Java internship at INTERNPE has been an immensely rewarding and educational experience. Throughout this internship, I have deepened my understanding of Java programming and its applications in building robust, scalable software solutions.

During my time at INTERNPE, I worked on various projects that allowed me to apply core Java concepts, including object-oriented programming, multithreading, exception handling, and collections framework. These projects challenged me to think critically and develop efficient, maintainable code.

A notable project I contributed to involved CONNECT 4 Game . This project required extensive use of Java's features and reinforced my knowledge of design patterns, data structures, and algorithms. It also provided me with practical experience in using development tools such as Maven, Git, and Jenkins, and frameworks like Spring and Hibernate.

The collaborative environment at INTERNPE enabled me to work closely with experienced developers, participate in code reviews, and gain insights into best practices in software development. I also had the opportunity to enhance my debugging and problem-solving skills, ensuring that my code met the highest standards of quality.

Overall, this internship has been a pivotal step in my professional journey. It has equipped me with the technical skills and confidence to take on complex programming challenges and contribute effectively to any software development team. I am grateful for the mentorship and support provided by my colleagues at INTERNPE and look forward to applying the knowledge and experience I have gained in my future career endeavors.