|  |
| --- |
|  |

****

**DEPARTMENT OF COMPUTER SCIENCE AND IT**

**JGI Knowledge Campus**

**Jayanagar 9th Block, Bangalore – 560069**

# Software Development and Java Programming

### INTERNSHIP REPORT

***Submitted by***

**HARSHITHA D**

USN: 23MCAR0178

***in partial fulfillment for the award of the degree of***

## MASTER OF COMPUTER APPLICATIONS

## General

**DEPARTMENT OF COMPUTER SCIENCE AND IT**

## 

**JGI Knowledge Campus**

**Jayanagar 9th Block, Bangalore – 560069**

This is to certify that the internship entitled

# Software Development and Java Programming

***is the bonafide record of Internship done by***

**HARSHITHA D**

**USN: 23MCAR0178**

**MCA – General**

during the year

**2023 -2025**

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Dr. Gobi N**  Guide  Department of Computer Science and IT  JAIN (Deemed-to-be University) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Dr. Murugan R**  Programme Head - MCA  Department of Computer Science and IT  JAIN (Deemed-to-be University) |

#### 

#### CERTIFICATE

This is to certify that  **Harshitha D** (Name), USN: **23MCAR0178**  for the programme of MCA in the Department of Computer Science and IT has fulfilled the Internship requirements prescribed for the MCA Programme in JAIN (Deemed-to-be University).

The Internship entitled, “ Software Development and Java Programming ” was carried out under my direct supervision. No part of the dissertation was submitted for the award of any degree or diploma prior to this date.

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Dr. Gobi N**  Guide  Department of Computer Science and IT  JAIN (Deemed-to-be University) |

**Internship Viva-voce:**

**Name of the Examiner Signature with Date**

1**. ........................................... ..........................................**

**2. ........................................... ..........................................**

#### 

#### DECLARATION

I affirm that the Internship work titled “ Software Development and Java Programming ” being submitted in partial fulfillment for the award of MASTER OF COMPUTER APPLICATIONS – [Specialization] is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University.

(Signature of the Candidate)

**Harshitha D** USN Number: 23MCAR0178

#### 

#### ACKNOWLEDGEMENT

I would like to acknowledge the following people who have encouraged, guided and helped to fulfill the Internship requirements prescribed for the MCA Programme in JAIN (Deemed to be University.

1. Mentor \_Dr Gobi N\_ for guiding me through pivotal moments of my study and professional career and for always being there to make sure that my progress was reviewed, documented and acknowledged. His / Her encouragement has been the greatest source of inspiration and confidence for carrying out my project work.
2. Finally, I would like to thank my family, to whom this work is dedicated, for their support and encouragement during these years.

**Special Thanks to:**

* Dr. Sagar Gulati, Director, School of Computer Science and IT, JAIN (Deemed-to-be University)
* Dr. Ananta Charan Ojha, Deputy Director, School of Computer Science and IT, JAIN (Deemed-to-be University)
* Dr. K Suneetha, Head, Department of Computer Science and IT, JAIN (Deemed-to-be University)
* Dr. MURUGAN R, Programme Head, MCA Programme, Department of Computer Science and IT, JAIN (Deemed-to-be University)





# EXECUTIVE SUMMARY

**Internship Overview**

The internship was a valuable learning experience that provided hands-on exposure to software development principles and practices. Through projects at Prodigy Infotech and CodSoft, I gained proficiency in problem-solving, algorithm development, and programming language. The internship fostered a strong foundation in object-oriented programming and user interface design.

**Key Learnings**

* **Problem-solving and Algorithmic Thinking**: Developing the temperature conversion program and guessing game at Prodigy Infotech honed my ability to break down complex problems into smaller, manageable steps and devise efficient algorithms.
* **Object-Oriented Programming (OOP):** Designing the ATM simulator and course management system at CodSoft reinforced my understanding of OOP concepts, including classes, objects, inheritance, and polymorphism.
* **User Interface (UI) Design:** Creating user-friendly interfaces for the ATM simulator enhanced my skills in designing intuitive and interactive user experiences.
* **Database Management**: Working on the course management system introduced me to database concepts and their application in managing and organizing data.

**Major Accomplishments**

* Successfully developed functional temperature conversion and guessing game programs.
* Implemented a robust ATM simulator with core banking functionalities.
* Demonstrated proficiency in Java programming and problem-solving techniques.

This internship served as a catalyst for my professional growth, equipping me with the necessary skills and knowledge to excel in the software development industry.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S .NO** | **TOPIC** | **PAGE NO** |
|  | **Abstract** | 03 |
| 1 | INTRODUCTION  1.1 About the Company  1.2 Background | 04 |
| 2 | SYSTEM ANALYSIS AND DESIGN  2.1 Task 1: Build a Temperature Conversion Program  2.2 Task 2 : Create a Guessing Game  2.3 Task 3 : Student Grade Calculator  2.4 Task 4 : ATM Interface | 05 |
| 3 | SYSTEM REQUIREMENTS  3.1 Hardware Specification  3.2 Software Specification | 13 |
| 4 | IMPLEMENTATION | 14 |
| 5 | RESULTS AND SCREENSHOTS | 18 |
| 6 | CONCLUSION  REFERENCES | 21 |

**Abstract**

This internship involved gaining practical experience in software development through projects at Prodigy Infotech and CodSoft.

At Prodigy Infotech, the focus was on building interactive programs. A temperature conversion program was developed to facilitate conversions between Celsius, Fahrenheit, and Kelvin scales, enhancing problem-solving and algorithmic skills. Additionally, a guessing game was created to improve logical reasoning and random number generation concepts.

At CodSoft, the internship centered on developing applications with practical utility. A student grade calculator was designed to automate the calculation of total marks, average percentage, and grades based on inputted subject marks. An ATM simulator was constructed to model real-world banking operations, emphasizing object-oriented programming principles and user interface design.

Overall, the internship provided a comprehensive learning experience, covering various programming paradigms and software development methodologies.

**Keywords:** software development, internship, programming, problem-solving, object-oriented programming, Java Programming.

**CHAPTER 1**

**INTRODUCTION**

* 1. **About the Company**

This internship was a dual experience, encompassing roles at both Prodigy Infotech and Codsoft.

**Prodigy Infotech** is a pioneering organization dedicated to revolutionizing education. With a mission to create accessible and innovative learning solutions, Prodigy Infotech empowers individuals of all backgrounds to reach their full potential. Their focus on providing tools and resources for students, professionals, and organizations aligns with the company's vision of transforming the learning landscape.

**Codsoft**, on the other hand, is a dynamic community fostering leadership development, learning, and engagement. By creating platforms for its members, Codsoft aims to cultivate a culture of shared interests and empower individuals to take on leadership roles. The company's emphasis on leadership development resonates with its commitment to driving positive change.

* 1. **Background**

The internship was a collaborative endeavor between Prodigy Infotech and Codsoft, offering a unique opportunity to gain exposure to both the dynamic world of education technology and the collaborative spirit of a community-focused organization. The primary focus at Prodigy Infotech was on software development, aligning with the company's mission of creating innovative learning solutions. The subsequent internship at Codsoft delved into Java programming, providing a deeper understanding of software development principles within a community-centric environment. This dual experience offered a well-rounded perspective on the software development industry and its applications across diverse sectors.

**CHAPTER 2**

**SYSTEM ANALYSIS AND DESIGN**

**2.1 Task 1: Build a Temperature Conversion Program**

The system requires a user to input a temperature value and its corresponding unit (Celsius, Fahrenheit, or Kelvin). The system should process the input, convert the temperature to the other two units, and display the results to the user.

**Key Requirements:**

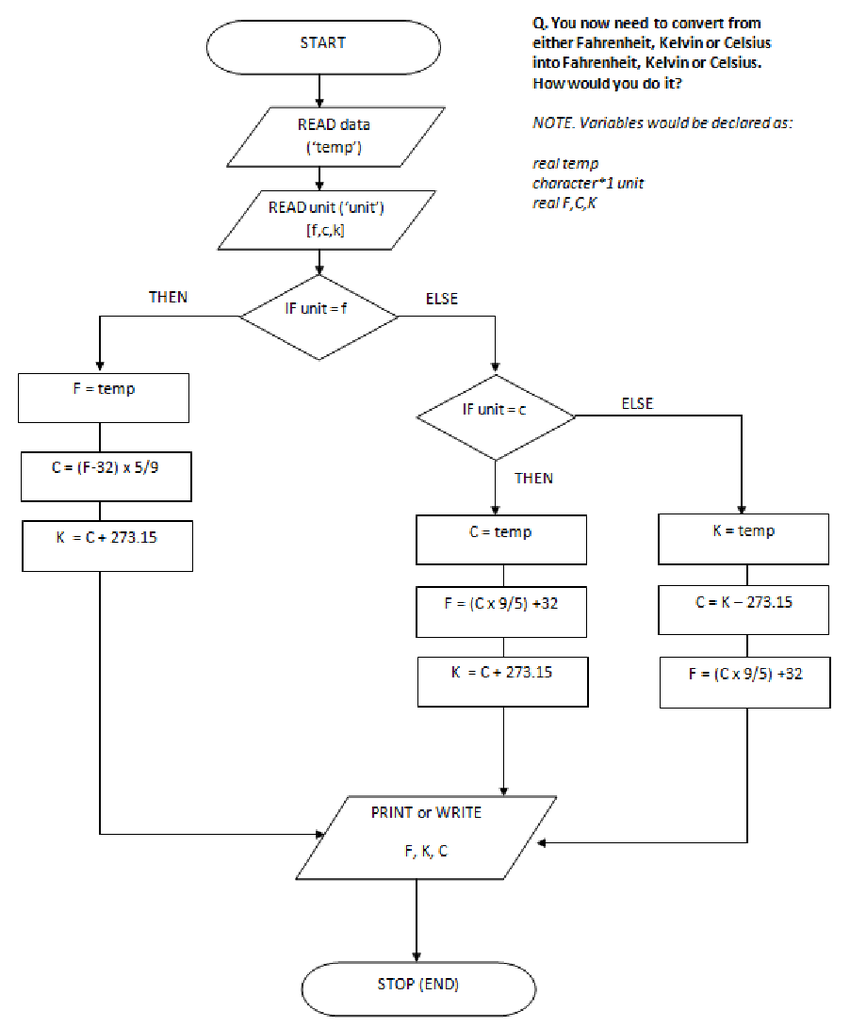
* **Input**: Temperature value and unit
* **Processing**: Temperature conversion calculations
* **Output:** Converted temperature values in all three units

**Components:**

* Input module: Prompts the user for temperature value and unit.
* Calculation module: Performs temperature conversions based on input unit.
* Output module: Displays converted temperatures in all three units.

**Process Flow:**

* Input module prompts the user to enter temperature value and unit.
* Input data is validated for correct format and unit.
* Calculation module determines the appropriate conversion formulas based on the input unit.
* Conversion calculations are performed to obtain temperatures in Celsius, Fahrenheit, and Kelvin.
* Output module displays the calculated temperature values in a user-friendly format.



The flowchart outlines a process for converting temperatures between Fahrenheit, Celsius, and Kelvin scales. The algorithm begins by inputting a temperature value and its corresponding unit. Based on the input unit, the flowchart directs the calculation to the appropriate conversion formula. For instance, if the input unit is Fahrenheit, the flowchart calculates the equivalent Celsius and Kelvin values. Similarly, if the input unit is Celsius or Kelvin, the other two temperature values are computed. Once all three temperature values are determined, they are printed or displayed as the output, and the process concludes.

**2.2 Task 2 : Create a Guessing Game**

The system requires the generation of a random number within a specified range. The user is then prompted to guess the number. The system compares the user's guess to the generated number and provides feedback. This process continues until the user correctly guesses the number. Finally, the system displays the number of attempts made by the user.

**Key Requirements:**

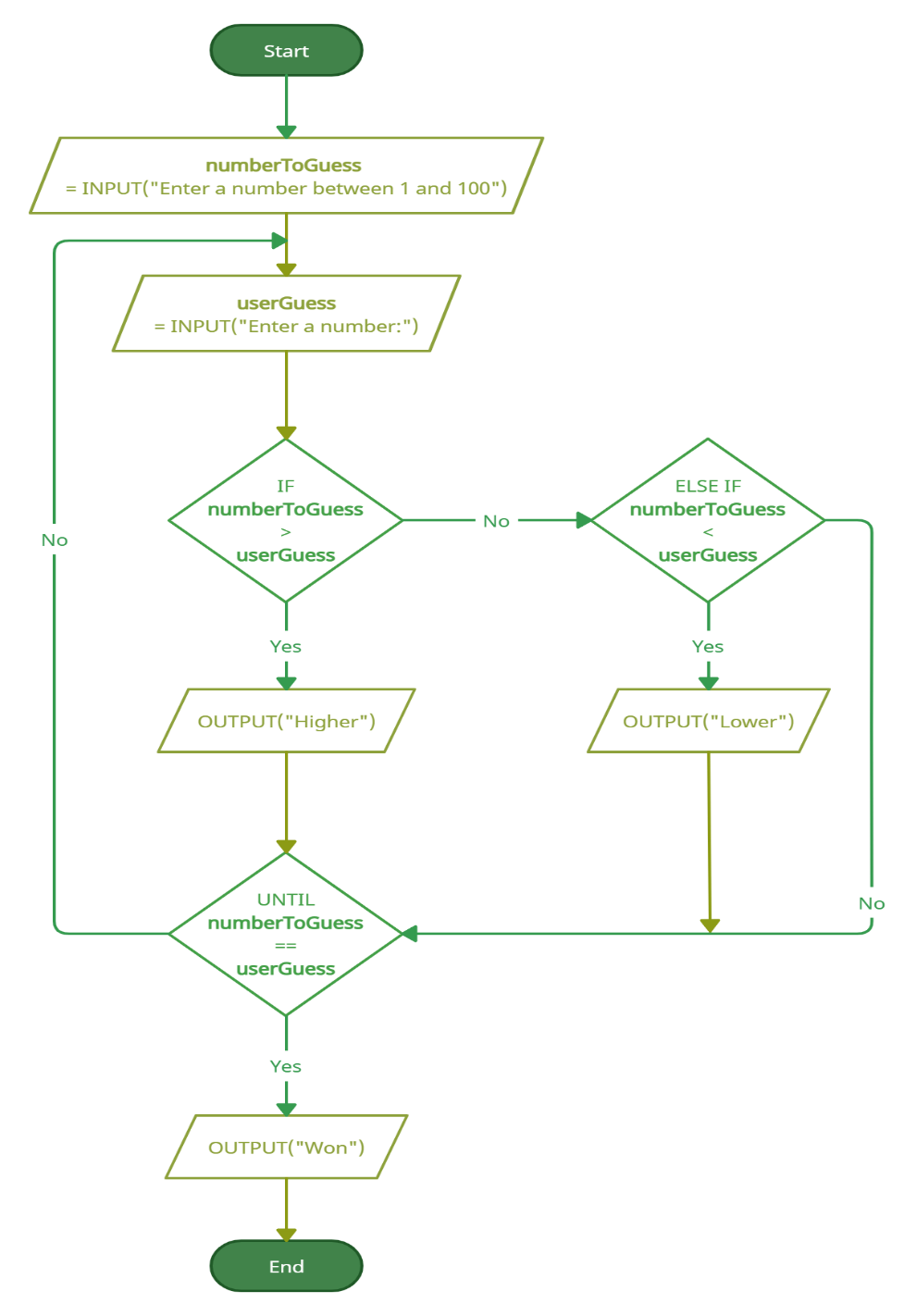
* Generate a random number.
* Prompt user for a guess.
* Compare guess with random number.
* Provide feedback (too high, too low, correct).
* Display the number of attempts upon correct guess.

**Components:**

* Random number generator: Generates a random number within a specified range.
* Input module: Prompts the user for a guess.
* Comparison module: Compares the user's guess with the generated number.
* Output module: Provides feedback and displays the number of attempts.
* Attempt counter: Keeps track of the number of guesses.

**Process Flow:**

1. Generate a random number.
2. Initialize an attempt counter to 0.
3. Prompt the user to enter a guess.
4. Increment the attempt counter.
5. Compare the user's guess with the generated number.
6. If the guess is correct, display the number of attempts and end the game.
7. If the guess is incorrect, provide feedback (too high or too low) and return to step 3.



The flowchart illustrates a number guessing game. The process begins by randomly selecting a number between 1 and 100. The user is then prompted to input their guess. The system compares the user's guess to the randomly generated number. If the guess is higher than the target number, the user is informed to guess lower. Conversely, if the guess is lower, the user is prompted to guess higher. This process continues in a loop until the user correctly guesses the number. Once the correct guess is made, the game ends, and the user is notified of their success.

**2.3 Task 3 : STUDENT GRADE CALCULATOR**

The system requires input of marks obtained by a student in multiple subjects. It calculates the total marks, average percentage, and assigns a grade based on the calculated percentage. The final results, including total marks, average percentage, and the corresponding grade, are displayed to the user.

**Key Requirements:**

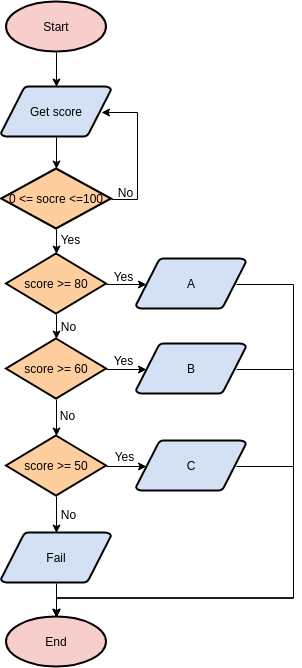
* Input: Marks obtained in each subject.
* Calculation: Total marks, average percentage, and grade determination.
* Output: Total marks, average percentage, and assigned grade.

**Components:**

* Input module: Collects marks for each subject.
* Calculation module: Computes total marks, average percentage, and assigns a grade.
* Output module: Displays the calculated results.

**Process Flow:**

1. Input marks for each subject.
2. Calculate the total marks by summing up all subject marks.
3. Calculate the average percentage by dividing total marks by the number of subjects.
4. Determine the grade based on the average percentage.
5. Display the total marks, average percentage, and assigned grade.



The flowchart outlines the process for calculating a student's grade based on their marks in multiple subjects. The process begins by inputting the marks obtained by the student in each subject. These marks are then summed up to calculate the total marks. The average percentage is determined by dividing the total marks by the number of subjects. Subsequently, the average percentage is used to assign a grade according to predefined grading criteria. Finally, the calculated total marks, average percentage, and the assigned grade are displayed as the output.

**2.4 Task 4 : ATM Interface**

The system simulates an ATM machine that provides basic banking functionalities. Users can perform transactions like withdrawing, depositing, and checking their account balance. The system interacts with a user's bank account to execute these transactions. Input validation and error handling are essential components of the system.

**Key Requirements:**

* User interface for displaying options (withdraw, deposit, check balance).
* Input handling for user selections and amounts.
* Interaction with a bank account for transaction processing.
* Output display for transaction results and account balance.
* Error handling for unexpected situations (insufficient balance, invalid input).

**Components:**

* ATM interface: Displays options, handles user input, and provides feedback.
* Bank account class: Manages account balance and transaction processing.
* Input validation module: Checks for valid input values.
* Error handling module: Handles unexpected situations and provides appropriate messages.

**Process Flow:**

1. Display ATM menu options (withdraw, deposit, check balance).
2. Accept user input for the desired option.
3. Validate user input.
4. If input is valid, execute the selected operation (withdraw, deposit, or check balance).
5. Update the bank account balance accordingly.
6. Display transaction result or error message.
7. Return to the main menu.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**3.1 Hardware Requirements**

* Processor: Intel Core i5 or equivalent
* RAM: 16 GB

**3.2 Software Requirements**

* Operating System: Windows 11
* Development Environment: Visual Studio
* Programming Language: Java

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 Temperature Converter Program**

**Problem: Temperature Conversion**

The Temperature Conversion Program is designed to facilitate efficient conversions between Celsius, Fahrenheit, and Kelvin temperature scales.

**Solution Approach**

**User Input and Validation** The program initiates by soliciting user input for the temperature value and its corresponding unit. A robust validation process is implemented to ensure the entered unit aligns with the accepted parameters (Celsius, Fahrenheit, or Kelvin). Upon successful validation, the program proceeds to the conversion phase.

**Conversion Logic** A dedicated convertTemperature function orchestrates the conversion process. This function employs conditional statements to accurately determine the appropriate conversion formula based on the input and target units. Once the correct formula is identified, the necessary calculations are performed to derive the equivalent temperature value in the desired unit.

**Output Generation** The program meticulously formats the converted temperature values for clarity and consistency. Each value is presented with two decimal places. In scenarios where the input and target units are identical, the program efficiently bypasses the conversion process and directly displays the original temperature value in all three units.

**Code Organization** To enhance code readability and maintainability, the program is structured into distinct modules. The main function serves as the program's entry point, handling user interactions, input validation, and invoking the convertTemperature function. Conversely, the convertTemperature function encapsulates the core conversion logic, promoting code reusability and facilitating potential modifications.

**4.2 Guessing Game Program**

**Problem: Guessing Game**

This program simulates a classic guessing game where the user attempts to guess a randomly generated secret number within a specified number of tries.

**Solution Approach**

**Random Number Generation:** The program utilizes the Random class to generate a secret number between 1 and 100 (inclusive). This ensures a fair and unpredictable guessing experience for the user.

**User Interaction and Input:** The program leverages the Scanner class to facilitate communication with the user. It displays instructions, prompts the user for their guess, and reads their input.

**Guess Evaluation and Feedback:** The program compares the user's guess to the secret number. It provides feedback based on the comparison:

* **Correct Guess:** If the user's guess matches the secret number, the program congratulates them and displays the number of attempts it took to guess correctly.
* **Guess Too Low:** If the user's guess is lower than the secret number, the program prompts them to try again, indicating that their guess is too low.
* **Guess Too High:** If the user's guess is higher than the secret number, the program prompts them to try again, indicating that their guess is too high.

**Loop Control and Termination:** A while loop is employed to control the guessing process. The loop continues as long as the number of attempts remains less than the maximum allowed (7 in this case). This allows the user to make multiple guesses until they successfully identify the secret number or exhaust their attempts.

**Outcome Handling:** The program considers two possible outcomes at the end of the game:

* **Successful Guess:** If the user successfully guesses the secret number, the program congratulates them and displays the number of attempts taken.
* **Unsuccessful Guess:** If the user runs out of attempts without guessing correctly, the program reveals the secret number and informs the user that they have exhausted their tries.

**Code Organization**

The program is structured into the main method, which encompasses the entire game logic. Within this method, separate code blocks handle random number generation, user interaction, guess evaluation, loop control, and outcome handling. This modular structure promotes code readability and maintainability.

**4.3 Student Grade Generation Program**

**Problem:** Develop a program to calculate a student's total marks, average percentage, and grade based on marks obtained in individual subjects.

**Solution Approach:**

* **User Input:** The program prompts the user to enter the number of subjects. It then utilizes a loop to iterate through each subject and collect the corresponding marks using the Scanner class.
* **Mark Storage:** An array is created to store the marks obtained in each subject. This array size is dynamically allocated based on the user-provided number of subjects, allowing for flexibility in the number of subjects considered.
* **Total Marks Calculation:** A loop iterates through the marks array and accumulates the individual marks into a variable named totalMarks.
* **Average Percentage Calculation:** The program calculates the average percentage by dividing the total marks by the number of subjects and casting the result to a double for decimal precision.
* **Grade Determination:** Conditional statements are employed to determine the grade based on the calculated average percentage. The grading scheme follows a typical format (A >= 90, B >= 80, C >= 70, D >= 60, F < 60).
* **Output:** The program displays the following information:
  + Total marks obtained by the student.
  + Average percentage calculated based on the total marks.
  + Grade assigned based on the grading criteria.

**Code Organization:**

The code is structured within the main method of the StudentGradeCalculator class. Specific code blocks handle user input, mark storage, total marks calculation, average percentage calculation, grade determination, and output generation. This modular structure promotes code readability and maintainability.

**4.3 ATM Interface Program**

**Problem Solution**

The program employs object-oriented principles to model the ATM system, encapsulating the core functionalities within distinct classes. The BankAccount class represents a user's account, managing the account balance and providing methods for deposit and withdrawal operations. The ATM class serves as the interface between the user and the bank account, handling user interactions and orchestrating the transaction process.

**Core Functionalities**

* **User Interaction:** The ATM interface presents a menu of options to the user, allowing them to select the desired transaction (withdraw, deposit, check balance, or exit). The system utilizes a Scanner object to capture user input and directs the program flow accordingly.
* **Transaction Processing:** Based on the user's choice, the ATM class interacts with the BankAccount class to execute the corresponding transaction. Input validation is performed to ensure the entered amount is valid (positive for withdrawals and deposits). Error handling mechanisms are implemented to address potential issues such as insufficient funds.
* **Output Generation:** The ATM provides informative feedback to the user, displaying transaction results, current balance, and error messages as needed.

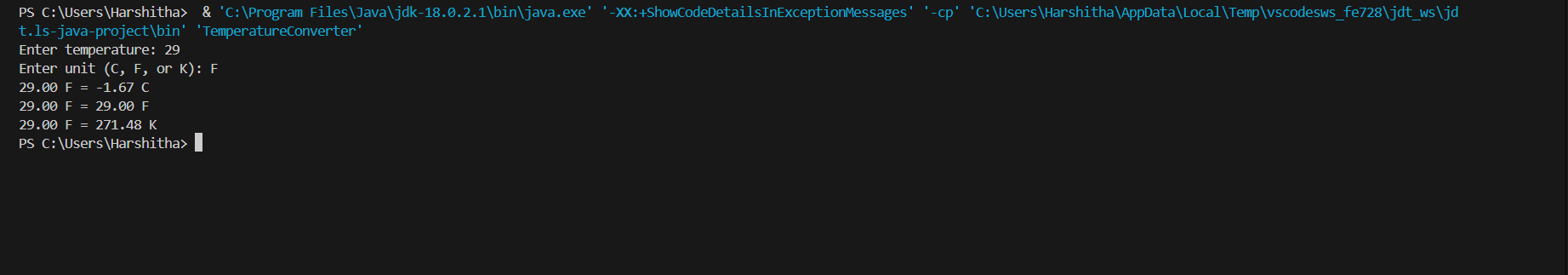
**Code Structure**

The code is organized into two primary classes: BankAccount and ATM. The BankAccount class encapsulates account-related data and operations, promoting data hiding and modularity. The ATM class manages user interactions, orchestrates the transaction process, and interacts with the BankAccount object. This object-oriented approach enhances code readability, maintainability, and reusability.

**CHAPTER 5**

**RESULTS AND SCREENSHOTS**

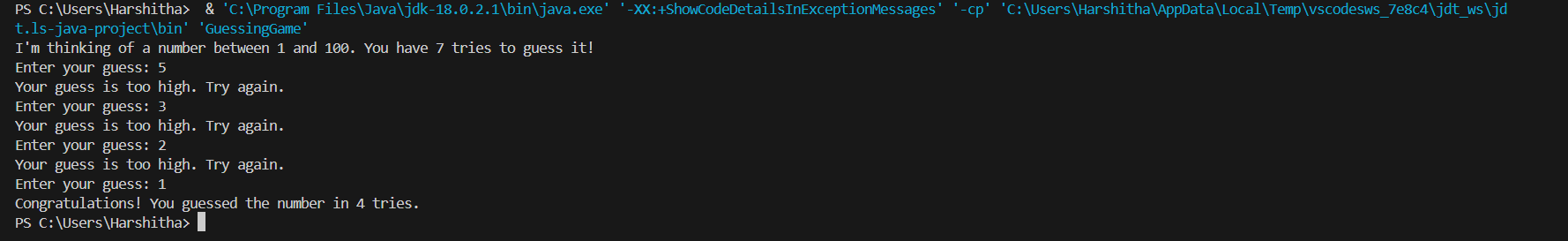
**5.1 Temperature Conversion Program Output**



The image depicts the output of a temperature conversion program.

The program successfully converts a temperature of 29 degrees Fahrenheit to its equivalent values in Celsius (-1.67 degrees) and Kelvin (271.48 degrees). The output is displayed in a clear and organized manner, presenting the original temperature, its unit, and the corresponding converted values in the other two units.

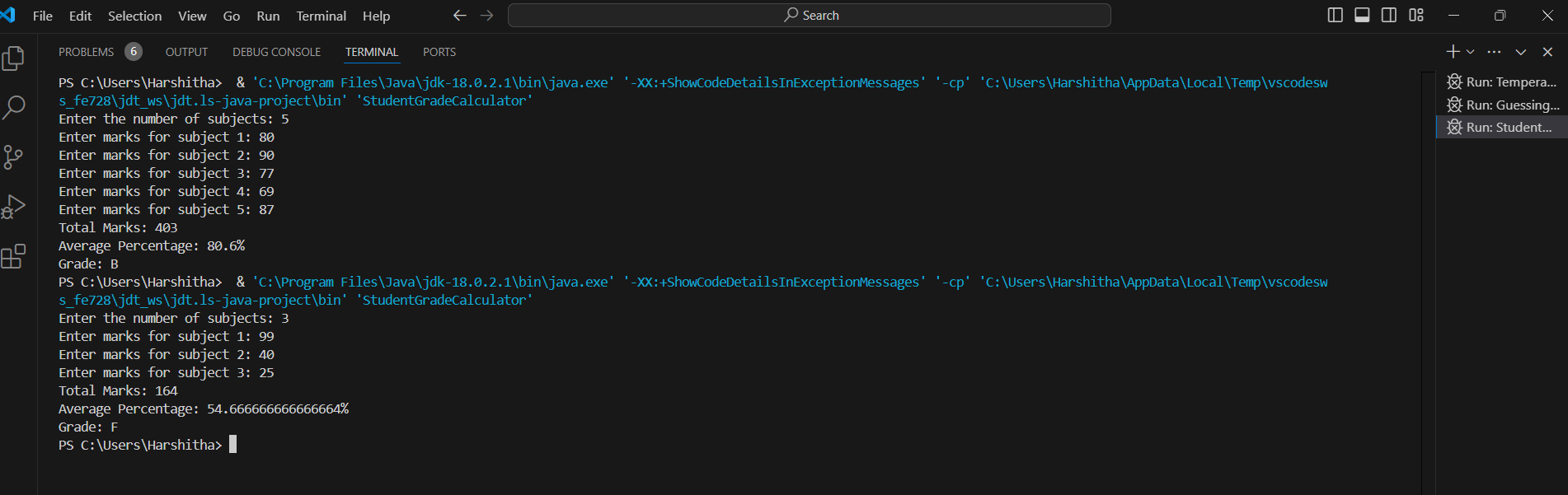
**5.2 Guessing Game Output**

****

The image depicts the output of a number guessing game.

The game begins by randomly selecting a number between 1 and 100 and informing the user that they have seven attempts to guess it. The user makes multiple guesses, and for each guess, the program provides feedback indicating whether the guess is too high or too low. Despite several attempts, the user fails to guess the correct number within the allotted tries. The game concludes by revealing the secret number.

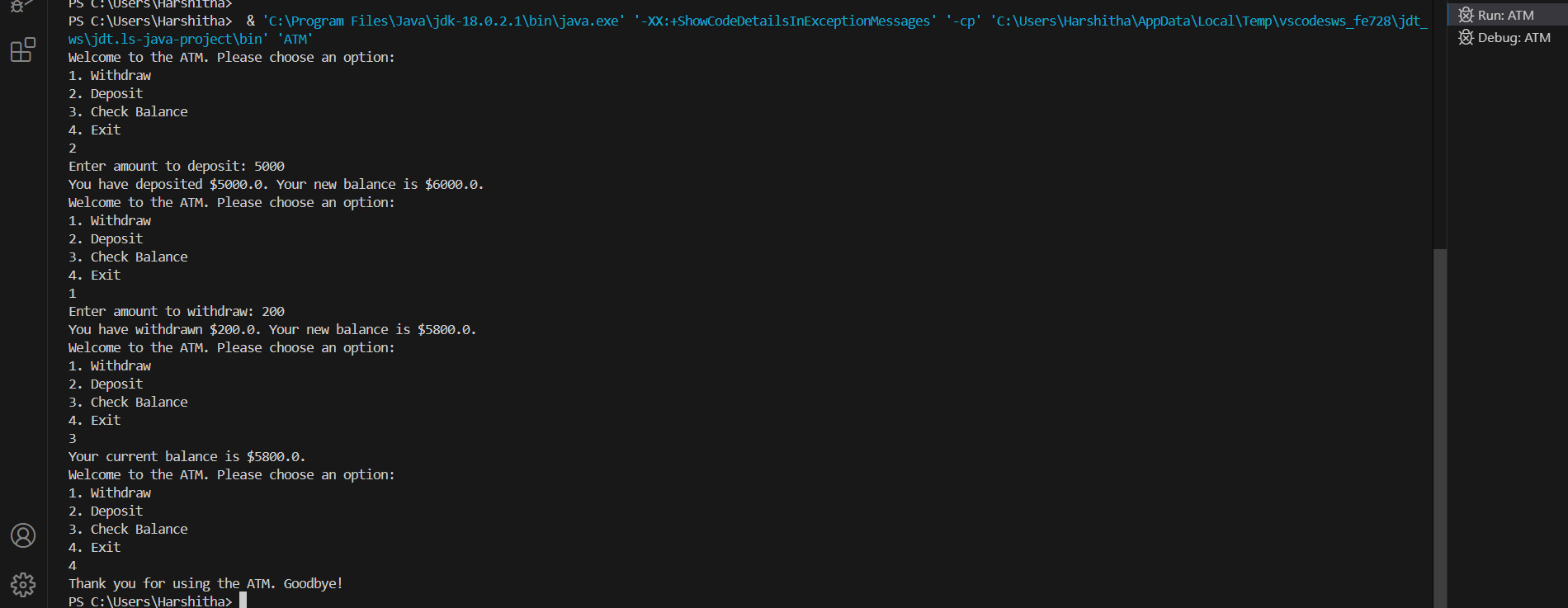
**5.3 Student Grade Calculator Output**

****

The provided image showcases multiple executions of the Student Grade Calculator program. The output primarily consists of user prompts, input values, and calculated results. The program begins by requesting the number of subjects, followed by individual subject marks input. Subsequently, it displays the calculated total marks, average percentage, and the corresponding grade based on a predefined grading scale.

The image demonstrates two distinct program runs with varying numbers of subjects and marks. The first run involves five subjects and results in an average percentage of 80.6% and a grade of B. The second run with three subjects yields an average percentage of 54.67% and a grade of F. These outputs highlight the program's ability to handle different input scenarios and accurately compute the desired results.

**5.4 ATM Interface Output**

****

The image depicts a simulated ATM interface, showcasing a basic transaction flow. The output demonstrates the core functionalities of an ATM, including deposit, withdrawal, and balance inquiry.

Upon execution, the ATM displays a welcome message and presents the user with a menu of options: withdraw, deposit, check balance, and exit. The user selects their desired action by entering the corresponding number.

In the provided example, the user first chooses to deposit $5000, and the ATM acknowledges the deposit, displaying the updated balance. Subsequently, the user withdraws $200, and the ATM deducts the amount from the balance, displaying the new balance. Finally, the user checks the balance to confirm the updated amount. The session concludes with the user selecting the exit option, and the ATM displays a farewell message.

The output demonstrates the ATM's ability to process basic transactions, provide appropriate feedback, and maintain accurate balance updates.

**CHAPTER 6**

**Conclusion**

The internship provided invaluable experience in software development and problem-solving. Through the development of the Temperature Conversion Program, Guessing Game, Student Grade Calculator, and ATM Interface, a strong foundation in programming fundamentals and object-oriented concepts was established.

The tasks offered opportunities to apply theoretical knowledge to practical applications, enhancing problem-solving abilities and critical thinking skills. The process of designing, coding, testing, and refining these programs contributed significantly to the overall learning experience.

While the internship focused on foundational programming concepts, it has laid the groundwork for exploring more complex software development challenges. The skills acquired during this period will be instrumental in future endeavors, enabling the application of these concepts to more intricate projects.

Overall, the internship was a rewarding experience that provided a solid base for a career in software development.

**Reference**

1. Java Tutorials: <https://docs.oracle.com/javase/tutorial/>
2. GitHub: <https://github.com/>
3. [visual-paradigm.com](https://online.visual-paradigm.com/diagrams/templates/flowchart/flowchart-example-temperature-to-celsius-conversion/)
4. [researchgate.net](https://www.researchgate.net/figure/Example-flow-chart-showing-the-algorithm-for-converting-between-temperatures_fig1_261613642)
5. [Guess the Number Game | Creately](https://creately.com/diagram/example/hrwbhxhm2/guess-the-number-game)
6. [ATM Workflow (Flowchart) - Software Ideas Modeler](https://www.softwareideas.net/a/395/atm-workflow-flowchart-)
7. **Tutorialspoint:** <https://www.tutorialspoint.com/java/>
8. **JavaTpoint:** <https://www.javatpoint.com/>
9. **HowToDoInJava:** <https://howtodoinjava.com/>
10. **Programiz:** <https://www.programiz.com/java-programming>
11. **GeeksforGeeks:** <https://www.geeksforgeeks.org/java/>