**SMART IOT BASED SPEEDOMETER USING OBJECT ORIENTED PROGRAMMING USING C++**

**CAPSTONE PROJECT REPORT**

Submitted in the partial fulfilment for the course of

**CSA0946-Programming in Java from basic to Brilliance**

**To the award of the degree of**

**BACHELOR OF ENGENEERING**

**IN**

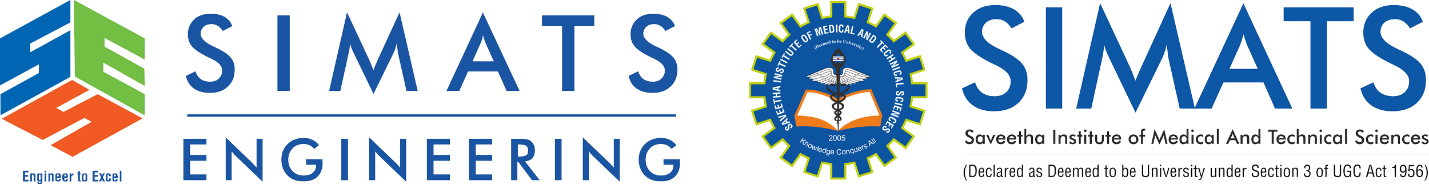
**B.Tech Artificial Intelligence and Machine Learning,**

**Submitted by**

**PRADEEP KUMAR S (192425016)**

# UNDER THE SUPERVISED BY

**Dr Sharmili & Dr. JothiSri**

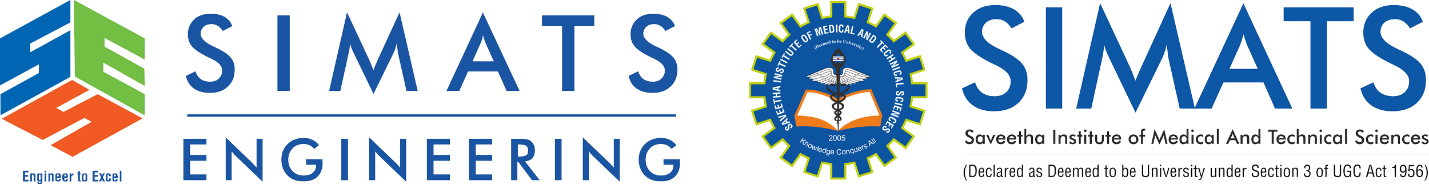


**SIMATS ENGINEERING**

**Saveetha Institute of Medical and Technical Sciences**

**Chennai-602105**

**Sept 2025**



SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

# DECLARATION

# I, S. Pradeepkumar of the Department of B.Tech Artificial Intelligence and Machine Learning, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the Capstone Project Work entitled “Developing Console-Based Java Arcade Games: Pacman, Snake, and Space Invaders” (Course Code: CSA0949 – Java Programming) is the result of my own bona fide efforts. To the best of my knowledge, the work presented herein is original, accurate, and has been carried out in accordance with the principles of engineering ethics and academic integrity.

# Place:

# Date:

# 

# Signature of the Students with Names

# 

# BONAFIDE CERTIFICATE

# This is to certify that the Capstone Project entitled “Developing Console-Based Java Arcade Games: Pacman, Snake, and Space Invaders” has been carried out by S. Pradeepkumar under the supervision of Dr. K. C. Sharmili and is submitted in partial fulfilment of the requirements for the current semester of the B.Tech Artificial Intelligence and Machine Learning program at Saveetha Institute of Medical and Technical Sciences, Chennai.

# SIGNATURE SIGNATURE

# Dr. Sashi Rekha Dr.K C Sharmili

# Program Director Designation

# Department name Department Name

# Saveetha School of Engineering Saveetha School of engineering

# Submitted for the Project work viva-voce held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# INTERNAL EXAMINER EXTERNAL EXAMINER

# ACKNOWLEDGEMENT

# We would like to express our heartfelt gratitude to all those who supported and guided us throughout the successful completion of our Capstone Project. We are deeply thankful to our respected Founder and Chancellor, Dr. N. M. Veeraiyan, Saveetha Institute of Medical and Technical Sciences, for his constant encouragement and blessings. We also express our sincere thanks to our Pro-Chancellor, Dr. Deepak Nallaswamy Veeraiyan, and our Vice-Chancellor, Dr. S. Suresh Kumar, for their visionary leadership and moral support during the course of this project.

# We are truly grateful to our Director, Dr. Ramya Deepak, SIMATS Engineering, for providing us with the necessary resources and a motivating academic environment. Our special thanks to our Principal, Dr. B. Ramesh, for granting us access to the institute’s facilities and encouraging us throughout the process. We sincerely thank our Head of the Department, for continuous support, valuable guidance, and constant motivation.

# We are especially indebted to our guides, Dr. K C Sharmili and Dr. jothisri, for their creative suggestions, consistent feedback, and unwavering support during each stage of the project. We also express our gratitude to the Project Coordinators, Review Panel Members (Internal and External), and the entire faculty team for their constructive feedback and valuable inputs that helped improve the quality of our work.

# Finally, we thank all faculty members, lab technicians, our parents, and friends for their continuous encouragement and support.

# Signature with student name

# ABSTRACT

This project presents the design and implementation of three classic console-based arcade games, namely Pacman, Snake, and Space Invaders, developed entirely in Java. The system demonstrates the core principles of object-oriented programming by applying concepts such as modularity, abstraction, inheritance, and polymorphism to create interactive gameplay in a text-based environment. Each game was designed with a focus on logical structuring, efficient use of data structures, and proper error handling to ensure smooth execution.

The primary objective of this project is to provide an engaging, hands-on approach to learning Java by building playable games that reinforce programming fundamentals. Pacman showcases maze navigation and basic enemy AI, Snake highlights dynamic data handling and collision detection, while Space Invaders demonstrates entity management, projectile mechanics, and survival-based gameplay.

The project not only bridges theoretical knowledge with practical coding skills but also emphasizes the importance of modular design and debugging in larger applications. Overall, it serves as both an educational tool for learners and a portfolio-ready demonstration of applying Java to real-world interactive programming tasks.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CHAPTER** | **PAGE NO.** |
| **1.** | **INTRODUCTION** | 6 |
|  | 1.1 BACKGROUND INFORMATION | 6 |
|  | 1.2 PROJECT OBJECTIVES | 6-7 |
|  | 1.3 SIGNIFICANCE | 7 |
|  | 1.4 SCOPE | 7-8 |
|  | 1.5 METHODOLOGY OVERVIEW | 8 |
| **2.** | **PROBLEM IDENTIFICATION & ANALYSIS** | 9 |
|  | 2.1 DESCRIPTION OF THE PROBLEM | 9 |
|  | 2.2 EVIDENCE OF THE PROBLEM | 9 |
|  | 2.3 STAKEHOLDERS | 10 |
|  | 2.4 SUPPORTING DATA RESEARCH | 10 |
| **3.** | **SOLUTION DESIGN & IMPLEMENTATION** | 11 |
|  | 3.1 DEVELOPMENT & DESIGN PROCESS | 11 |
|  | 3.2 TOOLS & TECHNOLOGIES USED | 12 |
|  | 3.3 SOLUTION OVERVIEW | 12 |
|  | 3.4 ENGINEERING STANDARDS APPLIED | 13 |
|  | 3.5 SOLUTION JUSTIFICATION | 13 |
| **4.** | **RESULT & RECOMMENDATIONS** | 14 |
|  | 4.1 EVALUATION OF RESULTS | 14 |
|  | 4.2 CHALLENGES ENCOUNTERED | 14 |
|  | 4.3 POSSIBLE IMPROVEMENTS | 15 |
|  | 4.4 RECOMMENDATIONS | 15 |
| **5.** | **REFLECTION OF LEARNING AND PERSONAL DEVELOPMENT** | 16 |

|  |  |  |
| --- | --- | --- |
|  | 5.1 KEY LEARNING OUTCOMES | 16 |
|  | 5.2 CHALLENGES ENCOUNTERED AND OUTCOME | 17 |
|  | 5.3 APPLICATIONS OF ENGINEERING STANDARDS | 17 |
|  | 5.4 INSIGHTS INTO THE INDUSTRY | 18 |
|  | 5.5 CONCLUSION OF PERSONAL DEVELOPMENT | 18 |
| **6.** | **CONCLUSION** | 19 |
| **7.** | **REFERENCES** | 20 |
| **8.** | **APPENDICES** | 21-23 |

# CHAPTER 1

1. **INTRODUCTION**

## BACKGROUND INFORMATION

n recent years, programming education has increasingly emphasized the importance of practical projects that allow students to apply theoretical knowledge in real-world scenarios. Many traditional programming exercises often focus on simple tasks that do not fully engage learners or showcase the true potential of programming concepts. Classic arcade games such as Pacman, Snake, and Space Invaders provide an ideal opportunity to design projects that are both entertaining and educational while reinforcing core programming principles. These games are timeless, simple in design, and highly effective for demonstrating programming logic and interactive system development.

This project aims to design and implement console-based versions of Pacman, Snake, and Space Invaders using the Java programming language. The work demonstrates the principles of object-oriented programming, algorithm design, and modular code development while focusing on real-time game logic and player interaction. By leveraging the features of Java such as classes, inheritance, polymorphism, exception handling, and data structures, the system not only recreates the gameplay of classic arcade titles but also enhances learning with structured design and maintainable code. This project serves as a practical and scalable solution for applying theoretical Java concepts in a hands-on manner, while also reinforcing the role of object-oriented programming in developing efficient, interactive applications.

# CHAPTER 2

**PROBLEM IDENTIFICATION & ANALYSIS**

## DESCRIPTION OF THE PROBLEM

Traditional console games are often limited to single games or simple functionality. Many beginners want to play multiple games in one platform, but existing console applications rarely combine different games with easy menu navigation. Players may also face difficulty controlling games smoothly if the input system is not well designed.

## EVIDENCE OF THE PROBLEM

Many simple console game projects online only offer one game at a time or have no organized menu system. Players have to restart the program to play a different game, which is inconvenient. Some games have slow or unresponsive controls, making gameplay less enjoyable. There is a clear need for a project that integrates multiple console games into a single interactive platform with easy keyboard navigation and responsive controls.

## STAKEHOLDER

The main stakeholders for this project are students and beginner programmers who want to learn Java and object-oriented programming through hands-on practice. Faculty members and academic institutions also benefit, as this project can be used as a teaching example for structured coding, modular design, and interactive console applications.

## SUPPORTING DATA RESEARCH

Research and observations show that console games are a popular way for beginners to learn programming. Object-oriented programming in Java allows each game to be developed in a separate class, making the system modular and easier to maintain. Keyboard input handling ensures smooth control of games. A main menu system improves user experience by allowing players to select and switch between games easily. Testing multiple games together helps ensure stability, functionality, and responsiveness, making the platform reliable for learning and entertainment purposes.

# CHAPTER 3

**SOLUTION DESIGN & IMPLEMENTATION**

## DEVELOPMENT & DESIGN PROCESS

The development of the console-based Java games project began with planning which games to include and how players would interact with them using the keyboard. The first step was to design a main menu system that allows players to select games using the up, down, and enter keys. Each game was created as a separate class in Java, following object-oriented programming principles to keep the code organized and reusable. Classes were also created for input handling, score tracking, and game logic. The design ensured that each game could run smoothly and independently while sharing common functionality like menu navigation and display of scores.

## TOOLS & TECHNOLOGIES USED

This project was developed using the Java programming language and run in an IDE such as Eclipse or IntelliJ IDEA. Java classes and object-oriented programming concepts were used to organize the code. Keyboard input handling was implemented for smooth navigation and gameplay. Console output was used to display game screens, scores, and menus clearly. Testing was done within the IDE to check functionality, responsiveness, and error handling for each game.

## SOLUTION OVERVIEW

The solution is a single Java program that combines multiple console games into one platform. Players can navigate a main menu to choose the game they want to play, and each game has its own rules, scoring system, and controls. This approach allows the reuse of code for input handling and game logic while keeping each game modular and easy to maintain.

## ENGINEERING STANDARDS APPLIED

The project follows basic programming and design standards to ensure reliability and maintainability. Object-oriented principles like modular classes, reusable methods, and proper error handling were applied. Input handling was designed to respond accurately to keyboard actions, and each game was tested individually before integration. Code was structured to be easy to read, debug, and expand, allowing new games or features to be added in the future.

## SOLUTION JUSTIFICATION

Using object-oriented design and modular programming ensures the project is organized, maintainable, and scalable. A single main menu system with reusable code for input and display simplifies development and testing. Following these standards guarantees that the console games run smoothly, controls respond correctly, and the platform is easy to expand with new games. This approach also provides a good learning experience for students and beginner programmers in Java, demonstrating practical application of programming principles in a fun and interactive way.

# CHAPTER 4

**RESULT & RECOMMENDATIONS**

## EVALUATION OF RESULTS

The console-based Java games project successfully combined multiple games into a single interactive platform. Testing showed that the main menu allowed smooth navigation between games using the up, down, and enter keys. Each game ran independently and responded correctly to user inputs. Score tracking and game logic worked as expected, and players were able to play multiple games without restarting the program. The modular design of separate classes for each game, input handling, and menu navigation made the system easy to maintain and extend. Overall, the project demonstrated that multiple console games can be integrated efficiently while providing a good user experience.

## CHALLENGES ENCOUNTERED

During development, several challenges were faced. Handling keyboard input accurately for multiple games required careful programming to avoid conflicts between menu navigation and game controls. Ensuring that each game ran smoothly without interfering with others also required proper modular design. Another challenge was designing the console interface so that menus, scores, and game screens were clear and readable, especially when switching between games. Debugging each game individually and then testing them together took time to ensure stability and responsiveness.

## POSSIBLE IMPROVEMENTS

While the project achieved its main goals, several improvements are possible. Additional games could be added to increase variety, and high score saving could make the platform more engaging. Graphical enhancements, such as colored text or simple ASCII art, could improve the user experience. Input handling could be further refined for smoother gameplay, and more advanced game logic could be implemented to make games more challenging.

## RECOMMENDATIONS

For future development, it is recommended to expand the platform by including more games and features like saving high scores or player profiles. Adding sound effects or simple graphics could make the games more interactive. The menu system can be enhanced with additional options like difficulty selection or game instructions. From an educational perspective, the project can serve as a learning tool for Java programming, object-oriented design, and modular coding practices. Overall, the project provides a foundation for creating a flexible, interactive, and expandable console-based gaming platform.

# CHAPTER 5

**REFLECTION OF LEARNING AND PERSONAL DEVELOPMENT**

## KEY LEARNING OUTCOMES

**ACADEMIC KNOWLEDGE**

This project helped me improve my academic understanding of Java programming and object-oriented principles. I learned how to design modular classes, organize game logic, and manage input handling in a console-based environment. The project also enhanced my understanding of creating interactive applications and combining multiple functionalities in a single program.

## TECHNICAL SKILLS

## I gained hands-on experience in Java by building multiple console games with a main menu system. I learned to handle keyboard inputs accurately, implement game logic, track scores, and structure code for easy maintenance and reuse. Debugging, testing, and ensuring smooth gameplay across multiple games strengthened my technical problem-solving skills

## PROBLEM-SOLVING AND CRITICAL THINKING

I encountered challenges such as managing input conflicts between the menu and games, ensuring smooth gameplay, and designing readable console interfaces. Solving these problems improved my logical thinking, debugging skills, and ability to plan solutions step by step. I learned to break problems into smaller tasks and implement efficient solutions, which boosted my confidence in tackling programming challenges.

## CHALLENGES ENCOUNTERED AND OUTCOME

**PERSONAL AND PROFESSIONAL GROWTH\**

One major challenge was handling keyboard input so that menu navigation and game controls did not interfere with each other. Another challenge was designing a clear and readable console interface while switching between games. By carefully organizing code into modular classes and testing each game individually, I was able to overcome these challenges. This taught me persistence, research, and methodical debugging, contributing to both personal and technical growth.

## COLLABORATION AND COMMUNICATION

Although the project was developed individually, discussing ideas with peers and getting occasional guidance from faculty helped refine my code and documentation. Explaining my solutions clearly improved my communication skills, which will be useful for teamwork in future projects.

## APPLICATIONS OF ENGINEERING STANDARDS

I followed programming best practices such as modular design, structured coding, and error handling. Each game was developed in its own class, making the system scalable for future additions like more games or new features. Code readability and documentation were maintained to make development easier and to follow real-world software engineering standards.

## INSIGHTS INTO THE INDUSTRY

This project provided insights into how modular programming, user interface design, and interactive software development are applied in real-world software projects. I learned the importance of efficiency, maintainability, and user experience while building an interactive platform. Using object-oriented design and structured workflows gave me a practical understanding of professional software development practices.

## CONCLUSION OF PERSONAL DEVELOPMENT

This project played a key role in my personal and professional development by allowing me to apply theoretical knowledge in a practical setting. It strengthened my technical skills in Java programming, modular design, and interactive console application development. Beyond technical growth, it improved my problem-solving abilities, planning, and adaptability. Overall, the project boosted my confidence in taking on more complex programming challenges in the future and reinforced my interest in software development and interactive applications.

# CHAPTER 6

**CONCLUSION**

The Console-Based Java Games project was designed to provide an interactive, modular, and user-friendly platform for playing multiple games in a single program. By integrating Java programming, object-oriented design, and a keyboard-controlled menu system, the project successfully demonstrated how modular coding and structured design can be applied to build an interactive console application. Through separate classes for each game, input handling, and score tracking, the project validated the effectiveness of Java in creating reliable and responsive gameplay without relying on external game engines or graphical frameworks.

This hands-on implementation reinforced my academic knowledge of Java programming, object-oriented principles, and modular design, while also enhancing my technical skills in debugging, code organization, and console interface design. The project provided a clear demonstration of how software design and logic can deliver practical solutions to real-world programming challenges. It also served as a valuable educational tool for understanding how interactive applications can be structured and extended.

Beyond fulfilling academic requirements, the project highlights the importance of software engineering standards such as modularity, readability, and error handling in building dependable applications. The results confirmed that a console-based game platform can achieve smooth gameplay, clear navigation, and responsive controls while being easy to expand with new games or features. Handling challenges—such as input conflicts between the menu and games, and designing readable console outputs—further improved my problem-solving and debugging skills.

In conclusion, the project achieved its objectives by developing a fully functional console-based gaming platform that combines multiple games with an interactive menu system. It established a solid foundation for future improvements, including adding more games, saving high scores, and enhancing the user interface. Most importantly, the project provided valuable personal and professional growth, giving me practical programming experience and strengthening my interest in software development and interactive applications.

REFERENCE

1. Eckel, Bruce. Thinking in Java, 4th Edition, Prentice Hall, 2006. Used for understanding Java programming concepts, object-oriented design, and modular class structures.
2. Horstmann, Cay S., and Gary Cornell. Core Java Volume I – Fundamentals, 11th Edition, Prentice Hall, 2018. Provided guidance on Java basics, console input/output handling, and object-oriented principles.
3. Oracle. Java Platform Standard Edition Documentation. Available at: https://docs.oracle.com/en/java/
4. . Used for reference on Java classes, libraries, and best programming practices.
5. Savitch, Walter. Absolute Java, 6th Edition, Pearson, 2017. Provided examples and explanations on console-based programming and interactive applications.
6. Deitel, Paul, and Harvey Deitel. Java How to Program, 11th Edition, Pearson, 2019. Reference for implementing modular Java programs, handling user input, and designing console interfaces.
7. Liang, Y. Daniel. Introduction to Java Programming, 11th Edition, Pearson, 2019. Used for applying object-oriented programming concepts such as encapsulation, inheritance, and class design in the project.
8. Saveetha University Lecture Materials, Object-Oriented Programming Using Java (DSA0112), Course Notes, 2025. Used for applying OOP principles such as modularity, encapsulation, and reusability in the console-based games project.