

# Pac-Man Game (Java)

## Abstract

This project is a Java-based Pac-Man game developed using Swing and AWT. The game includes multiple levels with different wall layouts, four intelligent ghosts, food collection mechanics, score tracking, and level progression. The player controls Pac-Man using keyboard inputs, aiming to eat all the food while avoiding ghosts. The game demonstrates core Java concepts such as object-oriented programming, event handling, GUI design, and game logic implementation.

---

## 1. Introduction

Pac-Man is a classic arcade game that involves navigating a maze, collecting food, and avoiding enemies. This project recreates the Pac-Man game using Java Swing and AWT, focusing on simplicity, clean logic, and interactive gameplay. It is designed as an academic and learning project to strengthen Java GUI and game development skills.

---

## 2. Objectives

- To design and develop a Pac-Man game using Java
  - To implement keyboard-based player movement
  - To create multiple levels with different wall layouts
  - To include four ghosts with random movement directions
  - To handle collision detection and game-over scenarios
  - To implement score calculation and level progression
  - To provide user options to retry or exit the game
- 

## 3. Technologies Used

- Programming Language: Java
  - GUI Framework: Java Swing and AWT
  - IDE: Eclipse
  - Concepts Used:
    - Object-Oriented Programming
    - Event Handling
    - Timers and Animations
    - Arrays and Game Maps
-

## 4. System Design

### 4.1 Game Components

- **Pac-Man:** Controlled by the player using arrow keys
- **Ghosts (4):** Move automatically with random directions
- **Walls:** Restrict movement
- **Food:** Increases score when collected
- **Levels:** Different maps for each level

### 4.2 Controls

- Arrow Keys: Move Pac-Man
  - ESC Key: Exit or continue the game
- 

## 5. Game Rules

1. Pac-Man must eat all food to complete a level
  2. If a ghost touches Pac-Man, the game ends
  3. On game over, the player can retry or exit
  4. Completing a level moves the player to the next level
  5. Each food collected increases the score
- 

## 6. Implementation Details

- The game uses a 2D array to represent the maze
  - Timer is used to control ghost movement
  - Collision detection checks Pac-Man and ghost positions
  - Dialog boxes are used for game over and exit confirmation
- 

## 7. Output Screens

- Game Window displaying maze, Pac-Man, ghosts, and score
  - Game Over dialog box
  - Level completion message
- 

## 8. Advantages

- Simple and interactive UI
- Demonstrates core Java concepts
- Easy to extend with more levels and features

---

## 9. Limitations

- Basic ghost AI (random movement)
- No sound effects
- Limited number of levels

---

## 10. Future Enhancements

- Smarter ghost AI
- Sound effects and background music
- Power pellets and special abilities
- High score storage
- More complex levels

---

## 11. Conclusion

The Pac-Man Java game project successfully demonstrates the implementation of a classic arcade game using Java Swing and AWT. It enhances understanding of GUI programming, event handling, and game logic. This project serves as a strong foundation for further improvements and advanced game development.

---

## Author

Saranya Durairaj BCA Graduate | Java Developer