Universidad Autónoma del Estado de México

IMPLEMENTATION OF OBJECT-ORIENTED PROGRAMMING CONCEPTS IN THE MINESWEEPER GAME USING PYTHON AND FLET

Celeste Chávez Martínez, Brayan Yañez Navarrete

Universidad Autónoma del Estado de México

Centro Universitario UAEM Atlacomulco

yaneznavarretebrayan@gmail.com

Abstract

This project presents the development of the classic Minesweeper game in Python using the Flet framework for the graphical user interface.
The objective was to apply key concepts of Object-Oriented Programming (OOP), including inheritance, polymorphism, abstract classes, and exception handling,
to design a functional and user-friendly application. The game provides three difficulty levels, ensures fair gameplay by placing mines after the first click,
and includes interactive elements such as flags, a mine counter, and real-time status updates. The implementation highlights how OOP principles support
modularity, reusability, and maintainability in software projects.

Introduction

The Minesweeper game is a classic puzzle that requires logical thinking and careful decision-making. Its implementation offers an ideal context
to apply OOP concepts in practice. By structuring the game with classes and objects, the system becomes easier to understand, extend, and maintain.
The use of Python and the Flet library allows the development of an interactive graphical interface that enhances the user experience.
The main challenge was to integrate OOP principles into the mechanics of Minesweeper while maintaining functionality and playability.

Methodology

The methodology followed consisted of designing the game around modular classes and applying OOP concepts systematically:

1. Inheritance: The MinesweeperGame class encapsulates the logic of the board, cells, and rules. Specific behaviors such as mine placement and cell revealing inherit common properties from the base structure.

2. Polymorphism: Functions like reveal() behave differently depending on the type of cell (mine, empty, or numbered). The use of conditional logic inside common methods demonstrates polymorphism, as the same action leads to different outcomes depending on the object's state.

3. Abstract Classes: While not fully implemented in the provided code, the design could be extended by defining an abstract Cell class using Python's abc module. This class would provide abstract methods like display() or reveal(), which subclasses (MineCell, EmptyCell, NumberCell) would implement differently.

4. Exceptions: Error handling is crucial in interactive applications. Custom exceptions can be raised when invalid moves occur, such as attempting to reveal a flagged cell or accessing coordinates outside the board. Python's try-except blocks ensure the game does not crash and provides meaningful messages to the user.

5. Graphical Interface (Flet): The interface was built with Flet, displaying difficulty selection, status messages, and instructions. User actions (left-click to reveal, long-click to flag) trigger event-driven functions that update the board in real-time.

Results

The final product is a fully functional Minesweeper game with three difficulty modes: easy (8×8 with 10 mines), medium (12×12 with 30 mines),
and hard (16×16 with 60 mines). The board is generated dynamically, ensuring the first click is always safe. The interface includes a mine counter,
real-time feedback, and clear visual instructions. Users can restart the game or exit through dedicated buttons. The implementation demonstrates the
power of OOP concepts when applied to real-world problems, resulting in clean, organized, and efficient code.

Conclusions

This practice confirmed that OOP principles are fundamental in software engineering. Inheritance and polymorphism allowed reusing and adapting code effectively.
Abstract classes provided a framework for designing extensible structures, while exceptions ensured robust execution. The integration of Flet for the graphical interface proved user-friendly and adaptable. Overall, the project reinforced the importance of applying OOP concepts in developing interactive applications, and it highlighted how such practices improve maintainability and scalability in programming projects.

References

- Python Documentation: https://docs.python.org/3/
- Flet Framework Documentation: https://flet.dev
- Grady Booch, Object-Oriented Analysis and Design with Applications, Addison-Wesley.