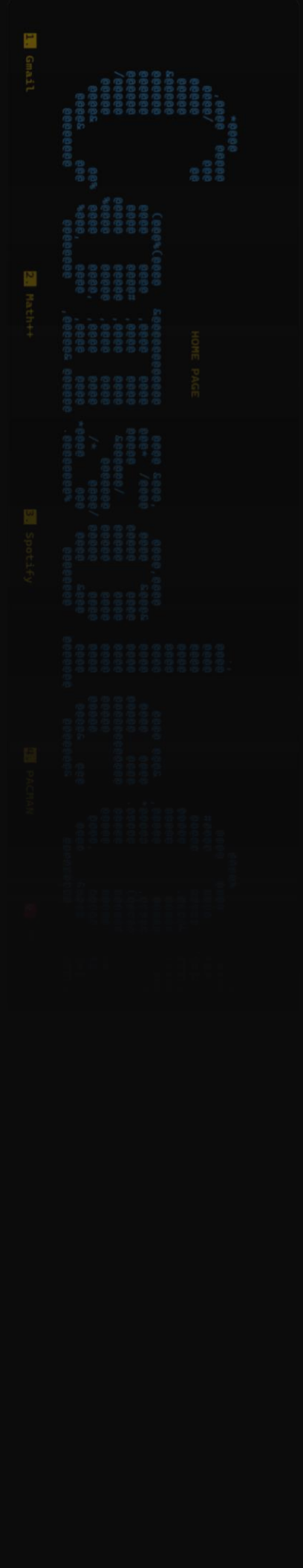


ConsoleOS

Object Oriented Programming
Project Report

ConsoleOS

by

SCHIZO ALLOY

OOP PROJECT REPORT

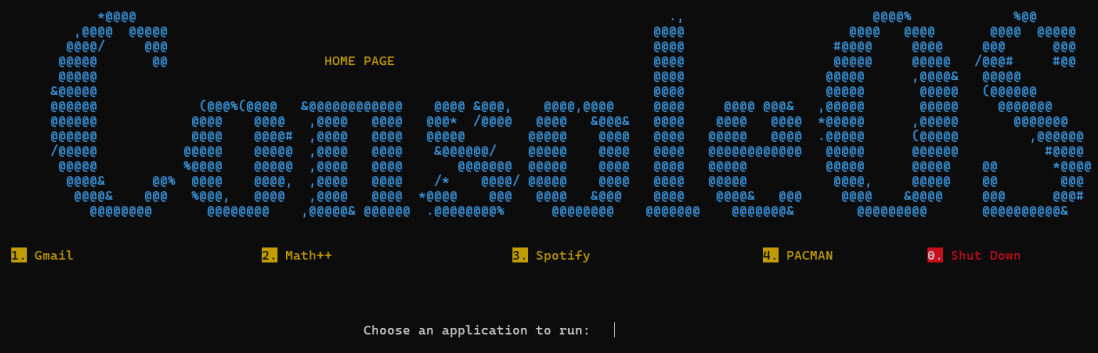
Contents

Abstract

This report explores the design and implementation of Console Operating System (ConsoleOS), a simple text-based operating system simulation. The project demonstrates various features and functionalities typical of an OS, utilizing object-oriented programming (OOP) concepts. Key features include a boot sequence, password protection, email application, game, and music player. The project leverages C++ and incorporates classes, inheritance, polymorphism, encapsulation, and abstraction to organize and manage system functionalities.

Introduction

ConsoleOS is a lightweight operating system simulation designed to run in a console environment. The primary objective is to demonstrate the application of OOP principles in creating a modular and extensible system. This project showcases a structured approach to building software systems, emphasizing reusability, scalability, and maintainability.



Features

Boot Screen Animation

- ConsoleOS begins with a welcoming boot screen animation, using ASCII art and timed delays to simulate an operating system startup sequence. This feature provides a visual and engaging introduction to the system.

Password Protection

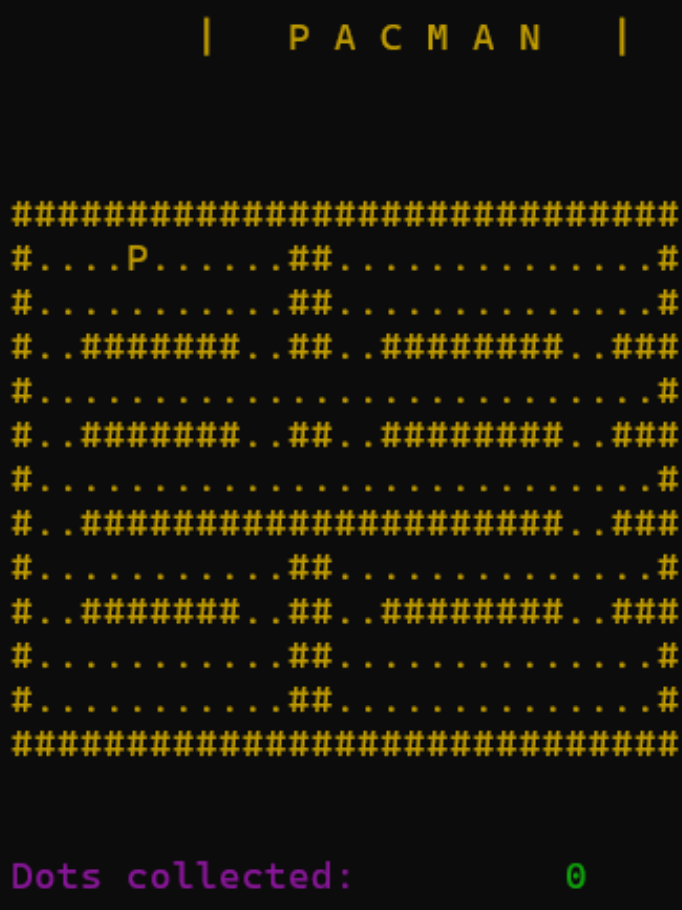
- To ensure security, ConsoleOS includes a lock screen that requires users to enter the correct password before accessing the system. Users are given a limited number of attempts to enter the correct password, with a hint provided after the maximum number of attempts is reached.

Email Application

- The email application within ConsoleOS allows users to send and view emails. Users can write new emails, view their inbox to check received messages, and view their outbox to see sent messages. The application simulates basic email functionalities, including message composition and storage. It utilizes text-file handling as a way to store the messages safely.

Game Application

- ConsoleOS includes a simple Pac-Man-like game where the player navigates a character on the screen, collecting dots while avoiding obstacles. This feature adds an element of entertainment and demonstrates the capability of the system to handle interactive applications.



Calculator Application

- The calculator application allows users to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. This application demonstrates the utility aspect of ConsoleOS, providing a functional tool for mathematical calculations.

Music Player

- The music player application enables users to play music files from the local drive. This feature showcases the ability of ConsoleOS to handle multimedia applications.

Shutdown Sequence

- Similar to the boot screen, ConsoleOS features a shutdown sequence with animations. This provides a visual indication that the system is shutting down, enhancing the user experience.

Object-Oriented Programming Concepts

Classes and Objects

- The project is structured around classes, each representing a different component or application within the system. For example, `Application` serves as a base class, with specialized applications such as EmailApp, Game, and MusicPlayer derived from it. Each class encapsulates data and methods relevant to its specific functionality.

Inheritance

- Inheritance is utilized to create a hierarchy of applications. The `Application` class is a base class from which other applications inherit. This allows common functionalities to be defined in the base class and specialized behaviors to be implemented in derived classes. For instance, EmailApp, Game, and MusicPlayer all inherit from Application.

Polymorphism

- Polymorphism is demonstrated through the use of virtual functions. The Application class defines a pure virtual function `run()`, which is overridden by each derived class. This allows the system to call the run() method on an Application pointer, and the correct method for the specific application (e.g., EmailApp::run(), Game::run()) will be executed. This enables different applications to be treated uniformly while exhibiting specific behaviors.

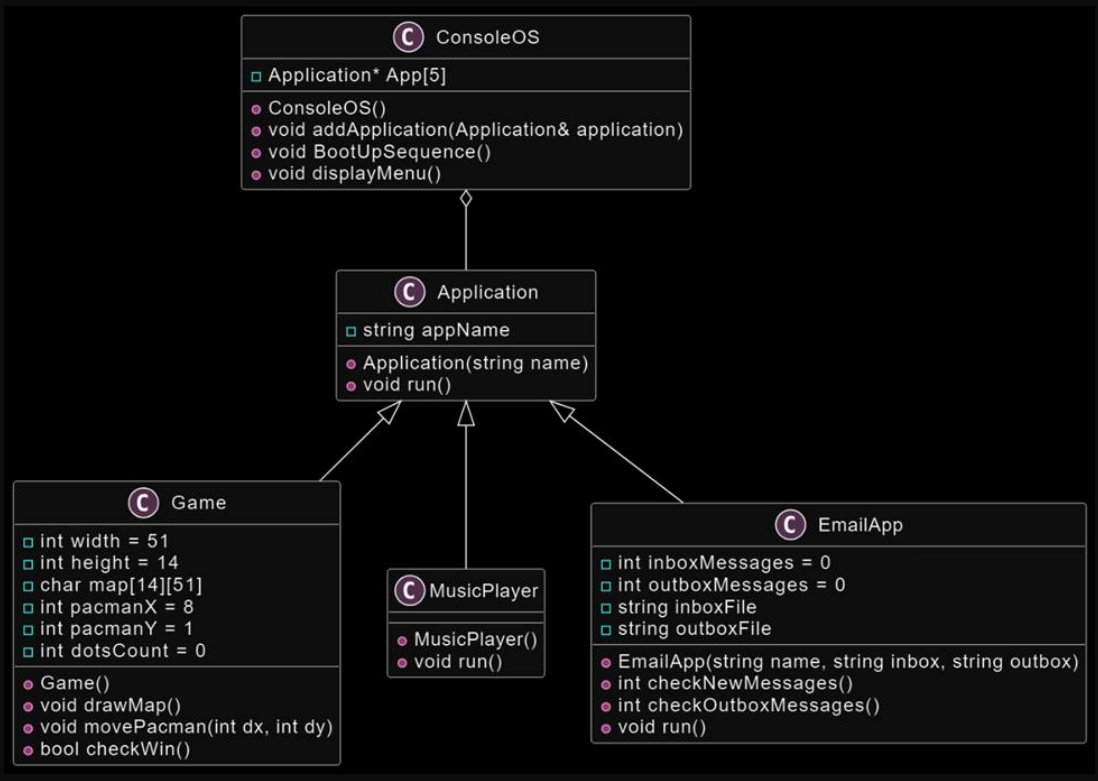
Encapsulation

- Encapsulation is achieved by hiding the implementation details of each class from other classes. For example, the EmailApp class manages its own email files and message counts internally, exposing only necessary methods to interact with the email functionalities. This ensures that the internal state of an object is protected from unintended interference.

Abstraction

- Abstraction is utilized to simplify complex functionalities and present them through simple interfaces. Users interact with high-level methods without needing to understand the underlying implementation details. For instance, the run() method in each application class abstracts the specific operations of that application, allowing users to start the application with a single method call.

UML Diagram:



Conclusion

ConsoleOS is a robust demonstration of object-oriented programming principles in a console-based environment. By implementing various features such as a boot screen, password protection, email application, game, and music player, the project showcases the practical application of OOP concepts. The use of classes, inheritance, polymorphism, encapsulation, and abstraction enables the creation of a modular, extensible, and maintainable system. This project serves as an educational tool for understanding and applying OOP in real-world scenarios.