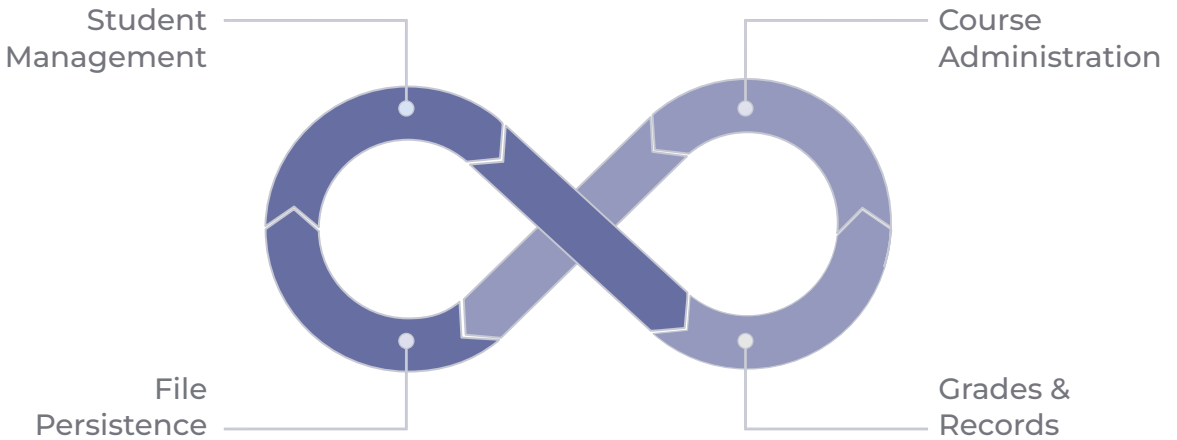
Campus Course & Records Manager (CCRM)

A comprehensive Java SE console application for managing students, courses, grades, and academic records. This project demonstrates advanced object-oriented programming principles, modern Java features, and robust file management capabilities.



Project Overview & Core Features

Student Management

Add, update, and track students with enrollment capabilities, profile generation, and transcript printing using Java Date/Time API.

Course Management

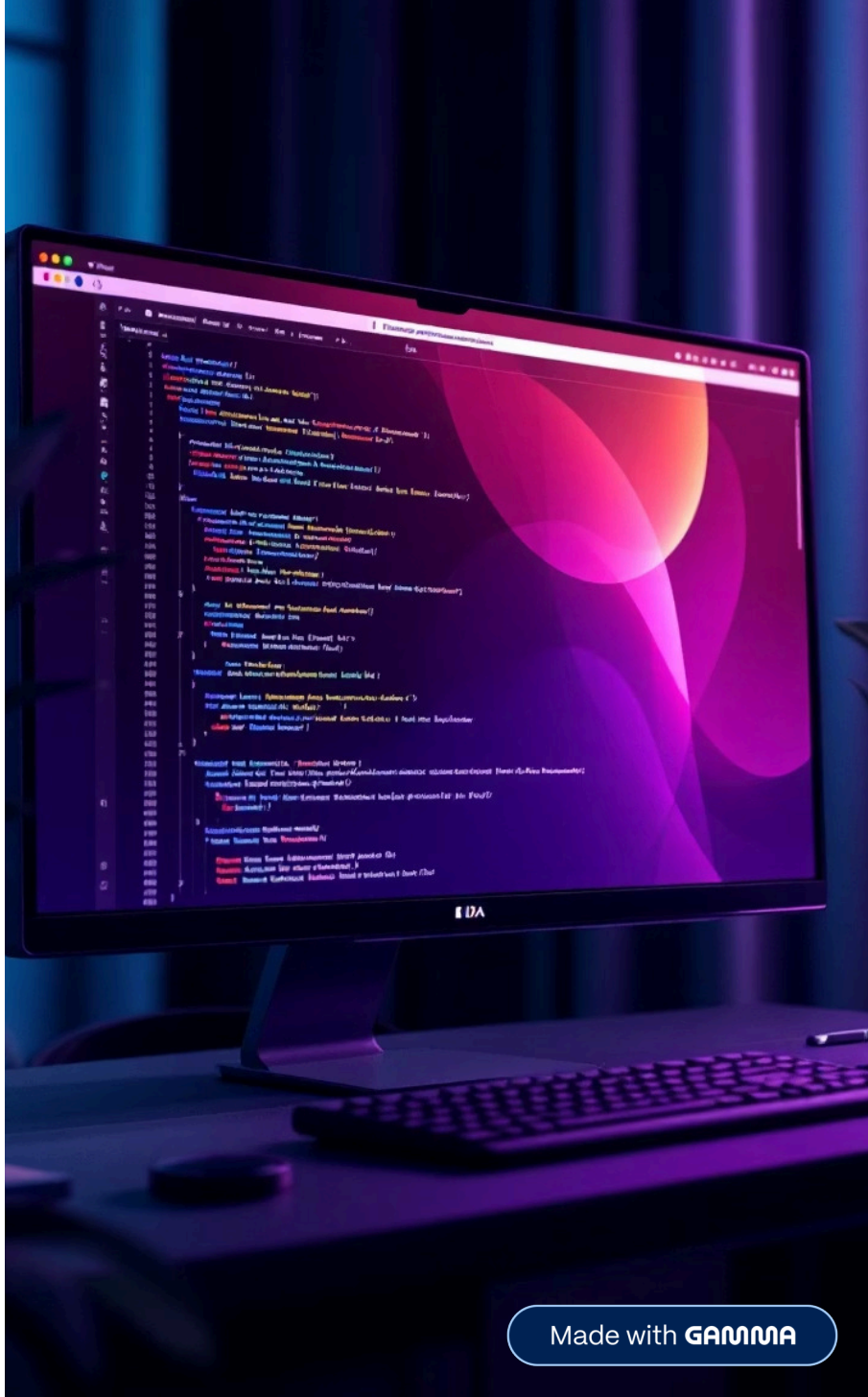
Create and manage courses with search and filter functionality using Stream API. Assign instructors and track departments.

Grading System

Record marks, compute GPA, and generate transcripts with enum-based grade system and semester management.

File Operations

Import/export CSV data, backup functionality with NIO.2, and recursive directory utilities for data management.



Java Platform Architecture

Understanding Java Components

JDK (Java Development Kit): Complete development environment including compiler, debugger, and tools for creating Java applications.

JRE (Java Runtime Environment): Runtime environment that provides libraries and JVM to run Java applications.

JVM (Java Virtual Machine): Platform-specific runtime engine that executes Java bytecode and manages memory.

Platform Editions

- **Java SE:** Standard Edition for desktop and server applications
- **Java EE:** Enterprise Edition for large-scale applications
- **Java ME:** Micro Edition for mobile and embedded devices

Technical Requirements & OOP Implementation

Encapsulation

Private fields with getter/setter methods ensure data integrity and controlled access to object properties.

Inheritance

Abstract Person class extended by Student and Instructor classes, demonstrating constructor chaining and super keyword usage.

Abstraction

Abstract classes and interfaces define contracts for implementation, hiding complex internal details from users.

Polymorphism

Virtual method invocation allows objects to behave differently based on their actual type at runtime.

Advanced Java Features Integration

Functional Programming

Lambda expressions for comparators, predicates for filtering, and functional interfaces for modern Java development.

Stream API

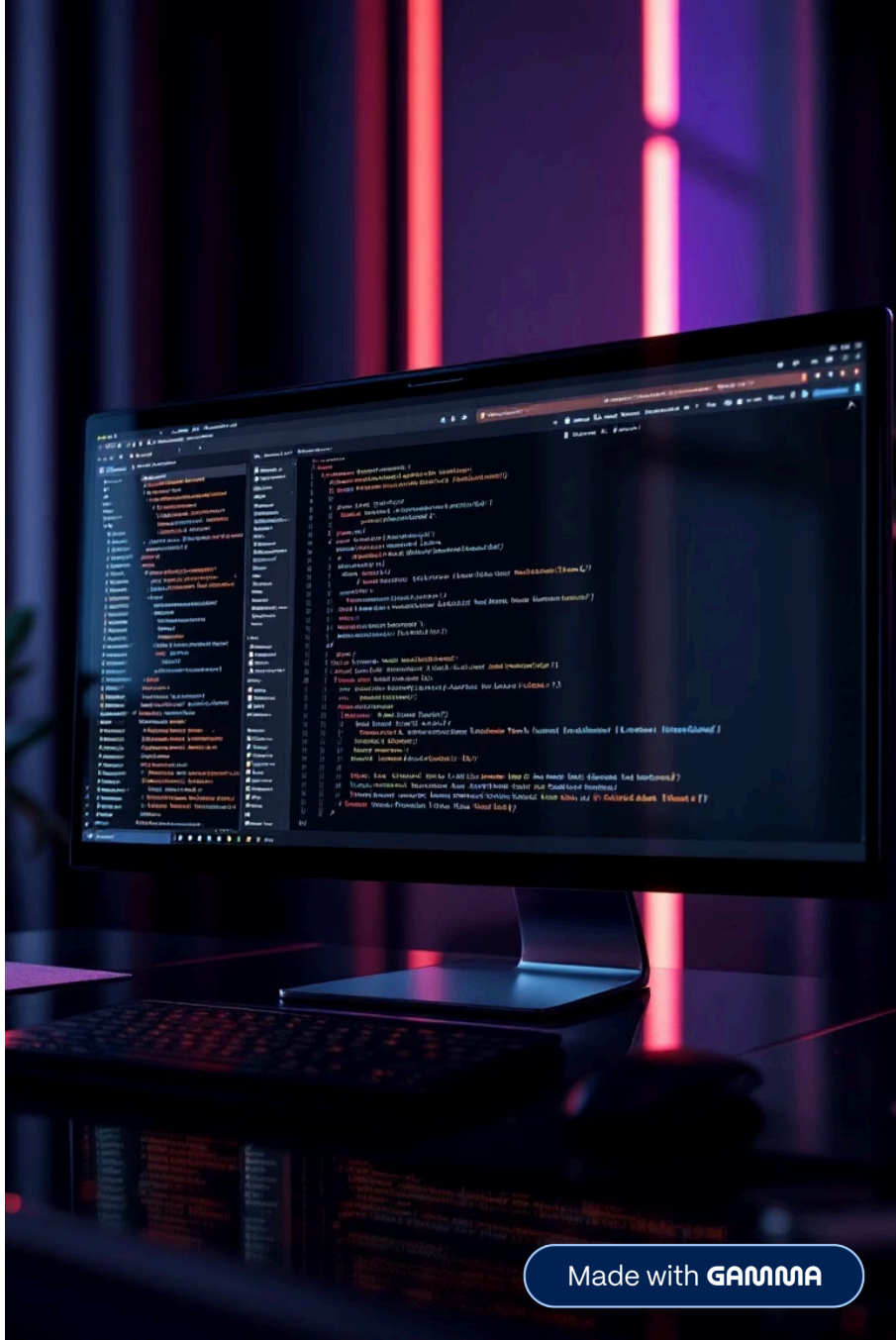
Process collections efficiently with filter, map, and reduce operations for course searches and GPA calculations.

Date/Time API

Modern temporal handling for enrollment dates, semester tracking, and backup timestamp generation.

NIO.2 File Operations

Path and Files APIs for robust file handling, CSV import/export, and recursive backup utilities.



Package Structure & Design Patterns

edu.ccrm.domain

Core entities: Person (abstract), Student, Instructor, Course, Enrollment with immutable value objects.

edu.ccrm.service

Business logic services with polymorphic interfaces for student, course, and transcript management.

edu.ccrm.io

File operations using NIO.2, CSV parsers, and backup services with recursive directory utilities.

edu.ccrm.cli

Console interface with switch-based menus and comprehensive user interaction handling.

Singleton Pattern

AppConfig class ensures single configuration instance across the application lifecycle.

Builder Pattern

Course.Builder and Transcript.Builder provide flexible object construction with method chaining.

Exception Handling & Data Validation

Custom Exceptions

DuplicateEnrollmentException and MaxCreditLimitExceededException provide specific error handling for business rules.

Multi-Catch Blocks

Comprehensive exception handling with try-catch-finally blocks for file operations and data processing.

Assertions

Runtime validation of invariants like non-null IDs and credit bounds with proper assertion configuration.

The application implements robust error handling strategies to ensure data integrity and provide meaningful feedback to users during all operations.

Development Environment Setup

01

Java Installation

Download and install JDK, configure PATH environment variables, and verify installation with java -version command.

02

Eclipse IDE Setup

Create new Java project, configure build path, set up run configurations, and organize package structure.

03

Project Configuration

Initialize Git repository, create README documentation, and set up test data files for import functionality.

04

Testing & Validation

Run application, test all features, capture screenshots, and create optional demo video for submission.

Project Deliverables & Submission

Git Repository

Complete source code with organized package structure, runnable main class, and comprehensive documentation.

Documentation

README.md with Java evolution timeline, platform comparisons, architecture explanations, and setup instructions with screenshots.

Demo Materials

Screenshots of installation, IDE setup, and program execution, plus optional video walkthrough demonstrating key features.

The final submission demonstrates mastery of Java SE fundamentals, advanced OOP concepts, modern language features, and professional software development practices. All technical requirements are mapped to specific implementations with clear documentation.