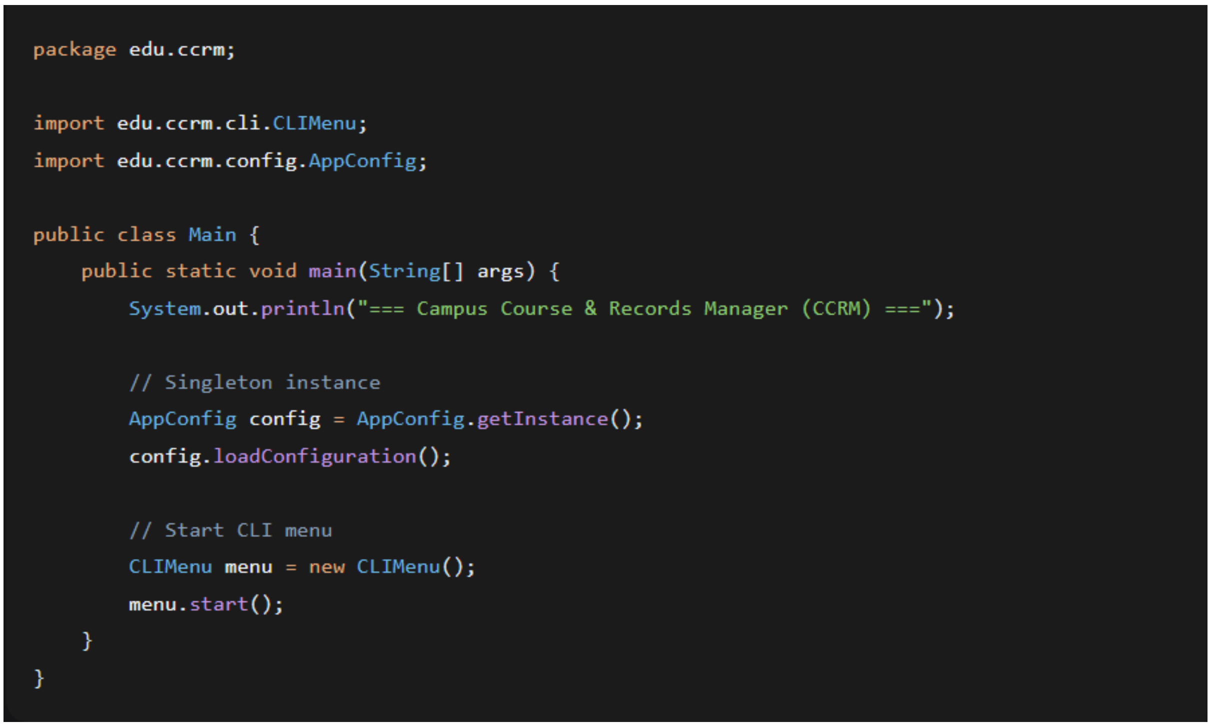
1.MAIN APPLICATION ENTRY POINT

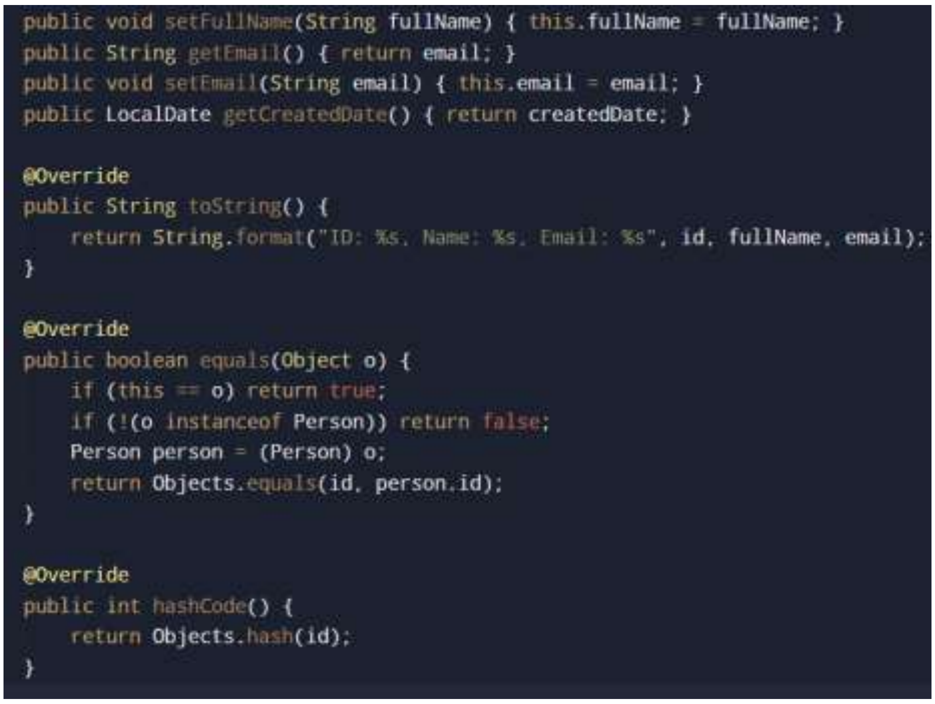
SRC/EDU/CCRM/MAIN.JAVA



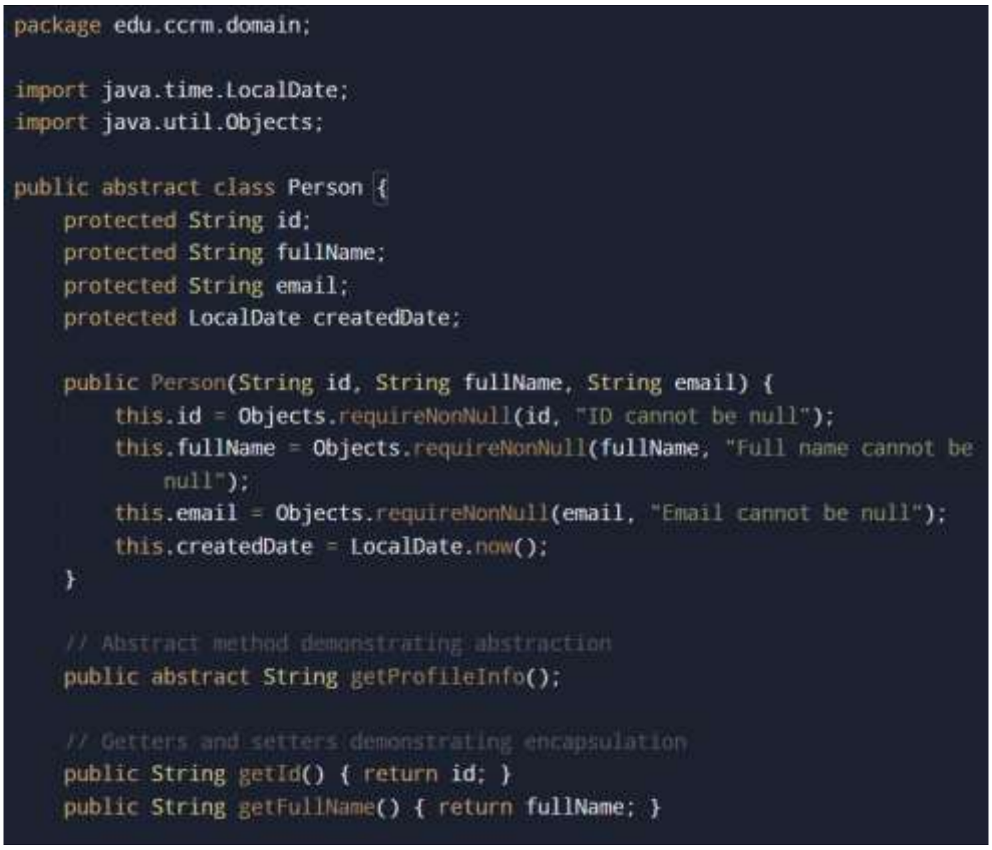
```java
package edu.ccrm;

import edu.ccrm.cli.CLIMenu;
import edu.ccrm.config.AppConfig;

public class Main {
    public static void main(String[] args) {
        System.out.println("=== Campus Course & Records Manager (CCRM) ===");

        // Singleton instance
        AppConfig config = AppConfig.getInstance();
        config.loadConfiguration();

        // Start CLI menu
        CLIMenu menu = new CLIMenu();
        menu.start();
    }
}
```

2.SINGLETON CONFIGURATION CLASS

src/edu/ccrm/config/appconfig.java



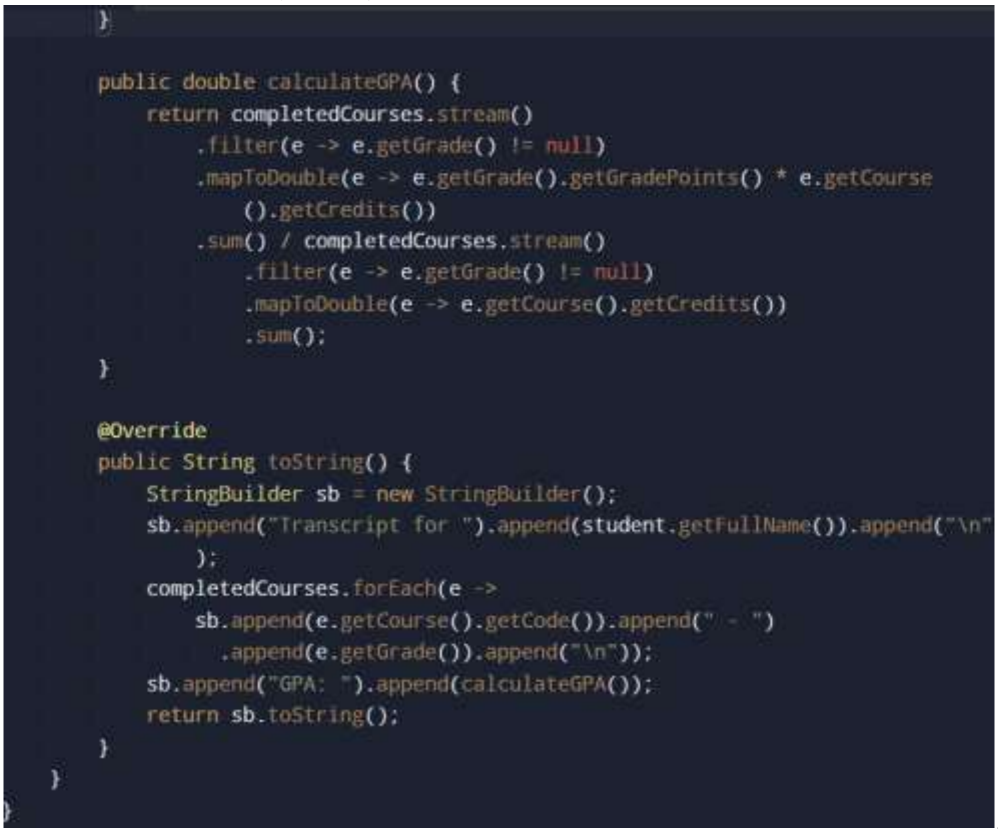
```java
public void setFullName(String fullName) { this.fullName = fullName; }
public String getEmail() { return email; }
public void setEmail(String email) { this.email = email; }
public LocalDate getCreatedDate() { return createdDate; }

@Override
public String toString() {
    return String.format("ID: %s, Name: %s, Email: %s", id, fullName, email);
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Person)) return false;
    Person person = (Person) o;
    return Objects.equals(id, person.id);
}

@Override
public int hashCode() {
    return Objects.hash(id);
}
```



```java
package edu.ccrm.domain;

import java.time.LocalDate;
import java.util.Objects;

public abstract class Person {
    protected String id;
    protected String fullName;
    protected String email;
    protected LocalDate createdDate;

    public Person(String id, String fullName, String email) {
        this.id = Objects.requireNonNull(id, "ID cannot be null");
        this.fullName = Objects.requireNonNull(fullName, "Full name cannot be
            null");
        this.email = Objects.requireNonNull(email, "Email cannot be null");
        this.createdDate = LocalDate.now();
    }

    // Abstract method demonstrating abstraction
    public abstract String getProfileInfo();

    // Getters and setters demonstrating encapsulation
    public String getId() { return id; }
    public String getFullName() { return fullName; }
```

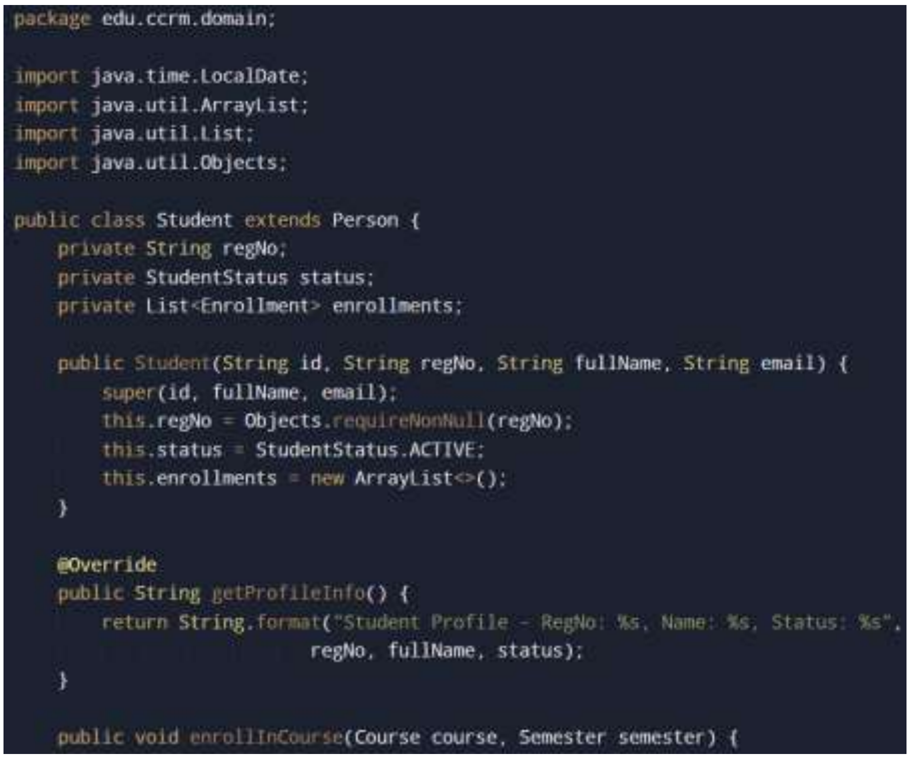
3.DOMAIN CLASSESS WITH OOP PRINCIPLE

Src/edu/ccrm/domain/person.java(Abstract Cla



```java
        }

        public double calculateGPA() {
            return completedCourses.stream()
                .filter(e -> e.getGrade() != null)
                .mapToDouble(e -> e.getGrade().getGradePoints() * e.getCourse
                    ().getCredits())
                .sum() / completedCourses.stream()
                    .filter(e -> e.getGrade() != null)
                    .mapToDouble(e -> e.getCourse().getCredits())
                    .sum();
        }

        @Override
        public String toString() {
            StringBuilder sb = new StringBuilder();
            sb.append("Transcript for ").append(student.getFullName()).append("\n"
                );
            completedCourses.forEach(e ->
                sb.append(e.getCourse().getCode()).append(" - ")
                  .append(e.getGrade()).append("\n"));
            sb.append("GPA: ").append(calculateGPA());
            return sb.toString();
        }
    }
}
```

ss)

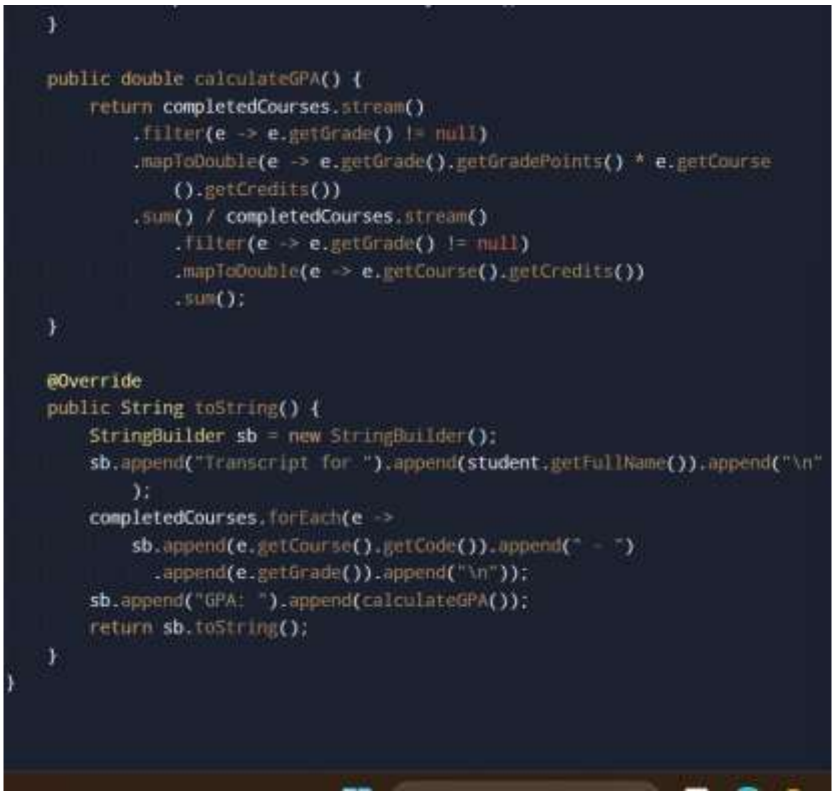


```java
package edu.ccrm.domain;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class Student extends Person {
    private String regNo;
    private StudentStatus status;
    private List<Enrollment> enrollments;

    public Student(String id, String regNo, String fullName, String email) {
        super(id, fullName, email);
        this.regNo = Objects.requireNonNull(regNo);
        this.status = StudentStatus.ACTIVE;
        this.enrollments = new ArrayList<>();
    }

    @Override
    public String getProfileInfo() {
        return String.format("Student Profile - RegNo: %s, Name: %s, Status: %s",
                           regNo, fullName, status);
    }

    public void enrollInCourse(Course course, Semester semester) {
```

Src/edu/ccrm /domain / student

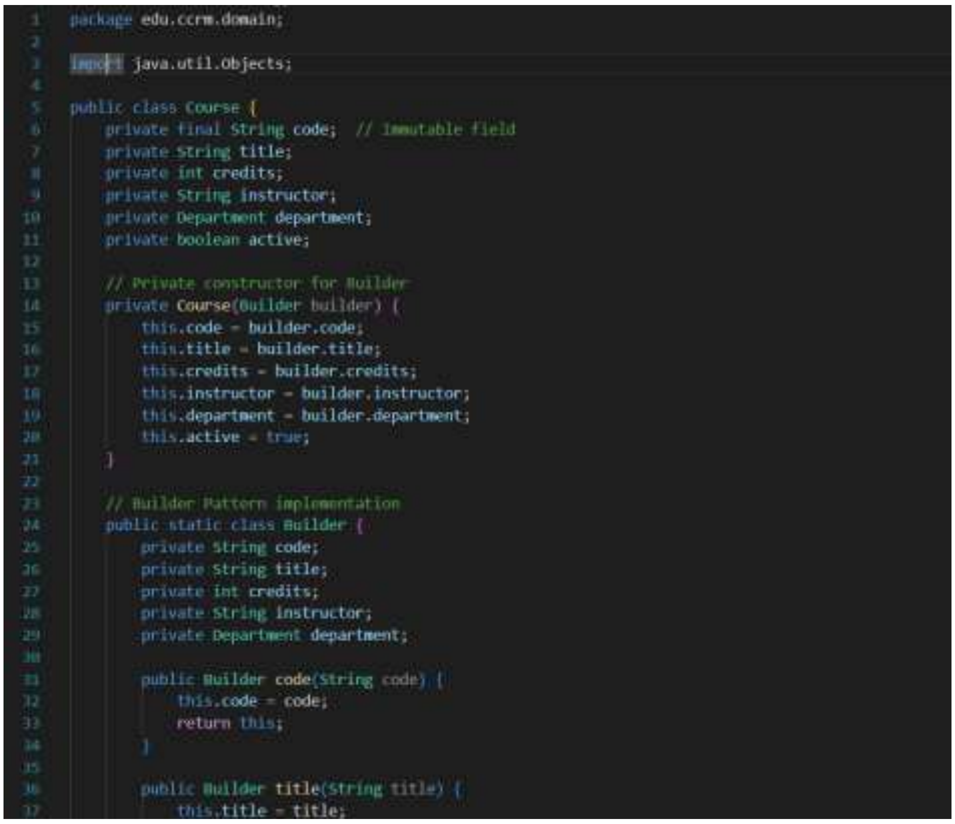
```java
package edu.ccrm.domain;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class Student extends Person {
    private String regNo;
    private StudentStatus status;
    private List<Enrollment> enrollments;

    public Student(String id, String regNo, String fullName, String email) {
        super(id, fullName, email);
        this.regNo = Objects.requireNonNull(regNo);
        this.status = StudentStatus.ACTIVE;
        this.enrollments = new ArrayList<>();
    }

    @Override
    public String getProfileInfo() {
        return String.format("Student Profile - RegNo: %s, Name: %s, Status: %s",
                           regNo, fullName, status);
    }

    public void enrollInCourse(Course course, Semester semester) {
```

s.java

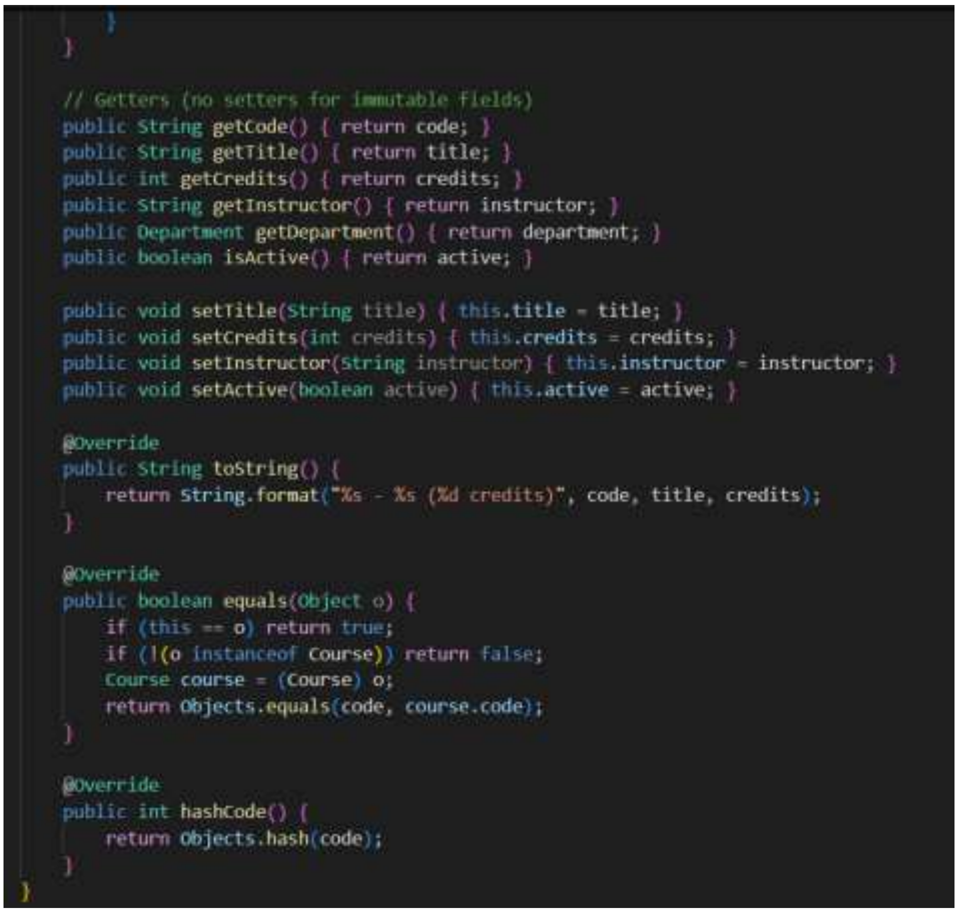


```java
    }

    public double calculateGPA() {
        return completedCourses.stream()
            .filter(e -> e.getGrade() != null)
            .mapToDouble(e -> e.getGrade().getGradePoints() * e.getCourse
                ().getCredits())
            .sum() / completedCourses.stream()
                .filter(e -> e.getGrade() != null)
                .mapToDouble(e -> e.getCourse().getCredits())
                .sum();
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Transcript for ").append(student.getFullName()).append("\n"
            );
        completedCourses.forEach(e ->
            sb.append(e.getCourse().getCode()).append(" - ")
              .append(e.getGrade()).append("\n"));
        sb.append("GPA: ").append(calculateGPA());
        return sb.toString();
    }
}
```

Src / edu / ccrm /domain/course.java(with Building Pattern)



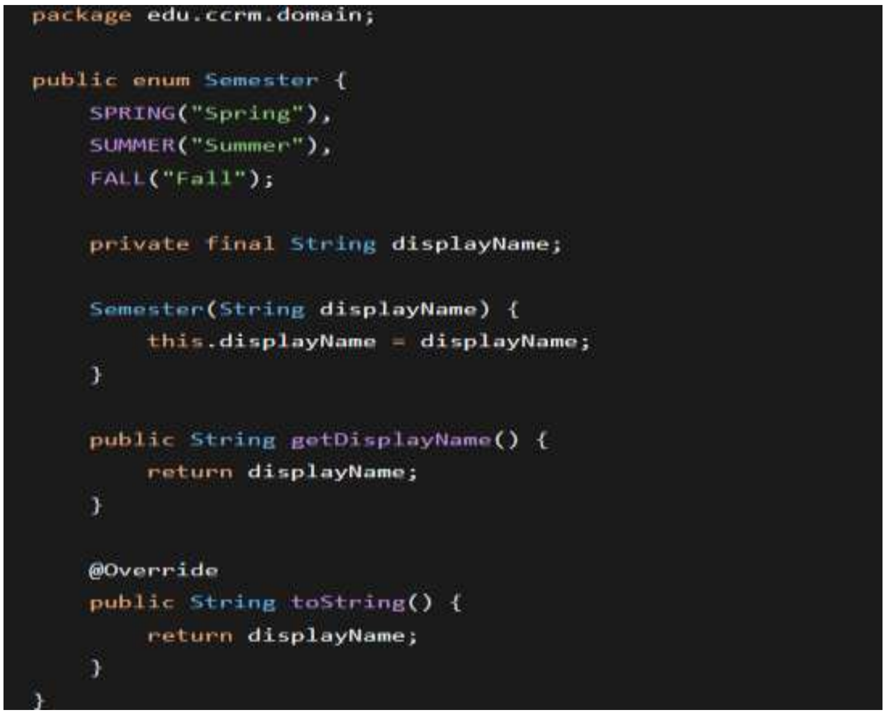
```java
package edu.ccrm.domain;

import java.util.Objects;

public class Course {
    private final String code;  // Immutable field
    private String title;
    private int credits;
    private String instructor;
    private Department department;
    private boolean active;

    // Private constructor for Builder
    private Course(Builder builder) {
        this.code = builder.code;
        this.title = builder.title;
        this.credits = builder.credits;
        this.instructor = builder.instructor;
        this.department = builder.department;
        this.active = true;
    }

    // Builder Pattern implementation
    public static class Builder {
        private String code;
        private String title;
        private int credits;
        private String instructor;
        private Department department;

        public Builder code(String code) {
            this.code = code;
            return this;
        }

        public Builder title(String title) {
            this.title = title;
```



```java
        }
    }

    // Getters (no setters for immutable fields)
    public String getCode() { return code; }
    public String getTitle() { return title; }
    public int getCredits() { return credits; }
    public String getInstructor() { return instructor; }
    public Department getDepartment() { return department; }
    public boolean isActive() { return active; }

    public void setTitle(String title) { this.title = title; }
    public void setCredits(int credits) { this.credits = credits; }
    public void setInstructor(String instructor) { this.instructor = instructor; }
    public void setActive(boolean active) { this.active = active; }

    @Override
    public String toString() {
        return String.format("%s - %s (%d credits)", code, title, credits);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Course)) return false;
        Course course = (Course) o;
        return Objects.equals(code, course.code);
    }

    @Override
    public int hashCode() {
        return Objects.hash(code);
    }
}
```

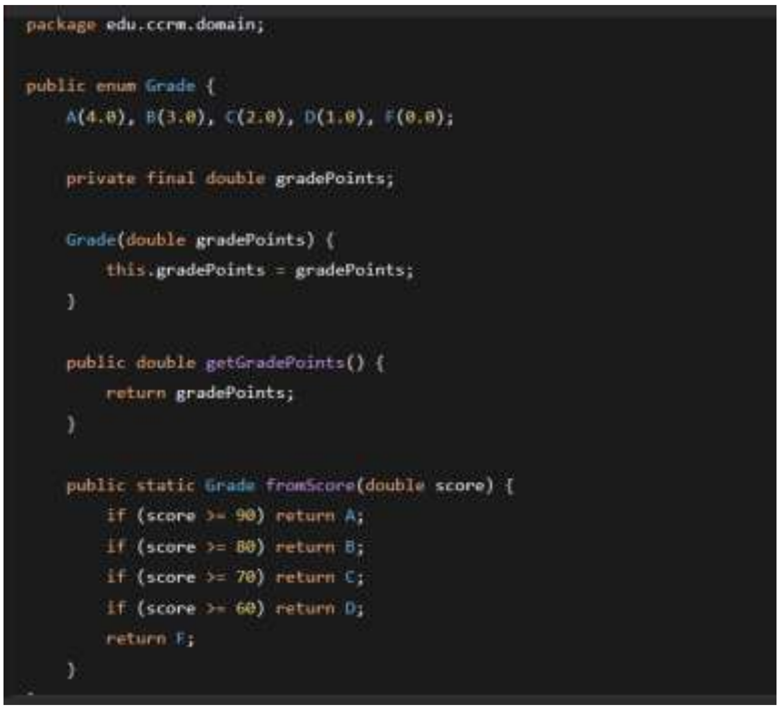
4. Enums

src/edu/ccrm/domain/Semester.java



```java
package edu.ccrm.domain;

public enum Semester {
    SPRING("Spring"),
    SUMMER("Summer"),
    FALL("Fall");

    private final String displayName;

    Semester(String displayName) {
        this.displayName = displayName;
    }

    public String getDisplayName() {
        return displayName;
    }

    @Override
    public String toString() {
        return displayName;
    }
}
```

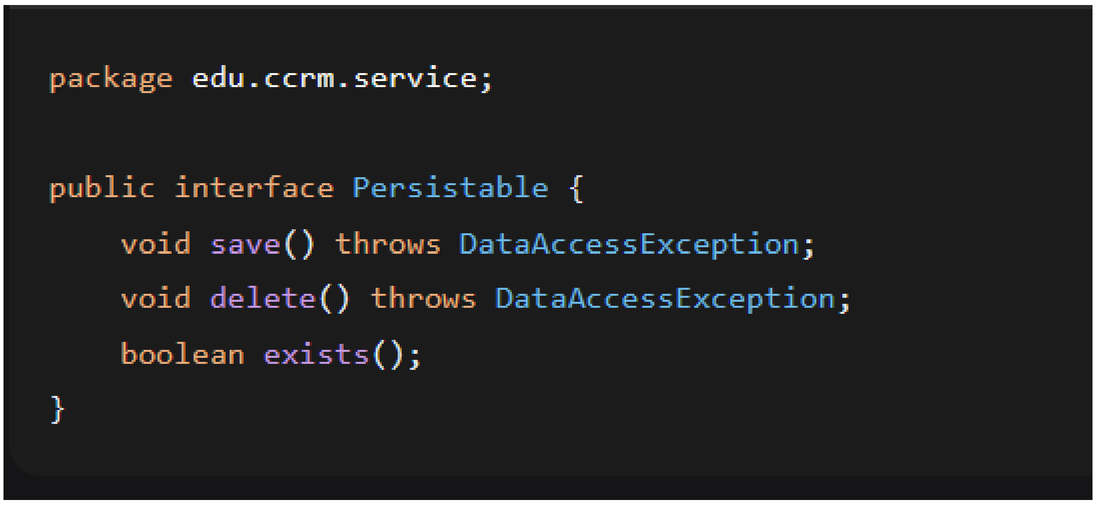
Src/edu/ccrm/domain/grade.java



```java
package edu.ccrm.domain;

public enum Grade {
    A(4.0), B(3.0), C(2.0), D(1.0), F(0.0);

    private final double gradePoints;

    Grade(double gradePoints) {
        this.gradePoints = gradePoints;
    }

    public double getGradePoints() {
        return gradePoints;
    }

    public static Grade fromScore(double score) {
        if (score >= 90) return A;
        if (score >= 80) return B;
        if (score >= 70) return C;
        if (score >= 60) return D;
        return F;
    }
```

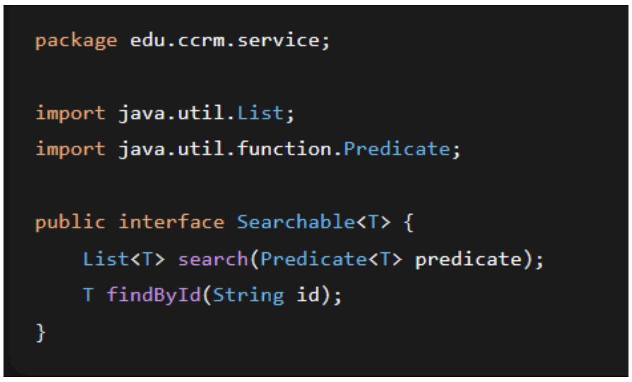
5.INTERFACES

Src/edu/ccrm/services/persistable.java



```java
package edu.ccrm.service;

public interface Persistable {
    void save() throws DataAccessException;
    void delete() throws DataAccessException;
    boolean exists();
}
```

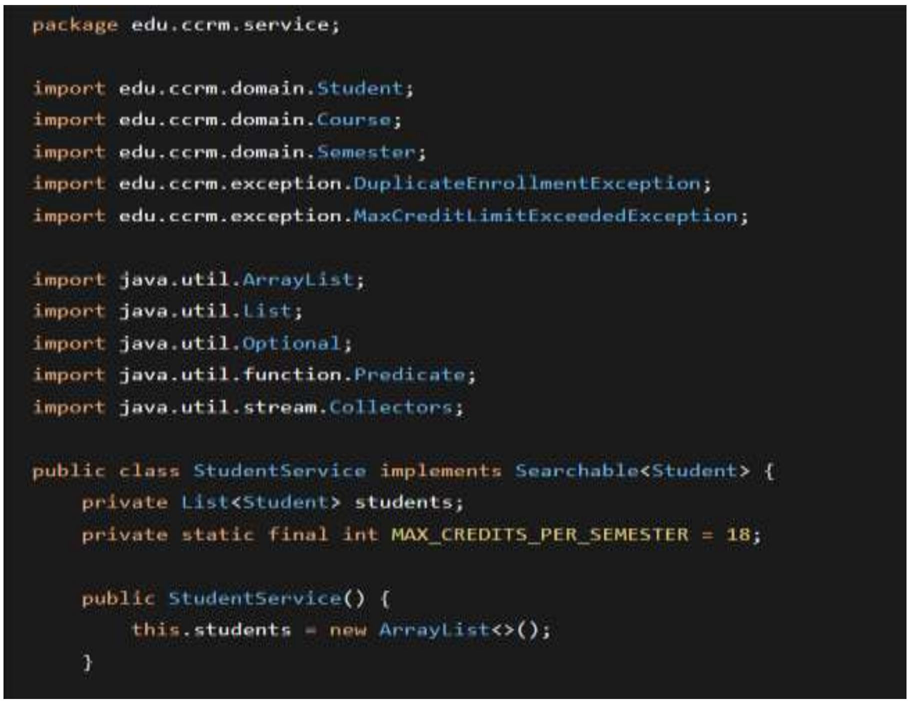
src/edu/ccrm/service/Searchable.java



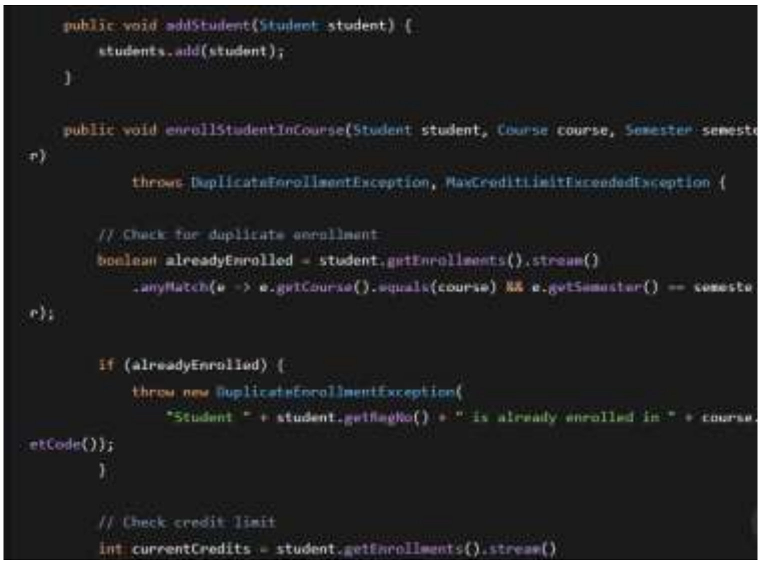
```java
package edu.ccrm.service;

import java.util.List;
import java.util.function.Predicate;

public interface Searchable<T> {
    List<T> search(Predicate<T> predicate);
    T findById(String id);
}
```

6. Service Classes

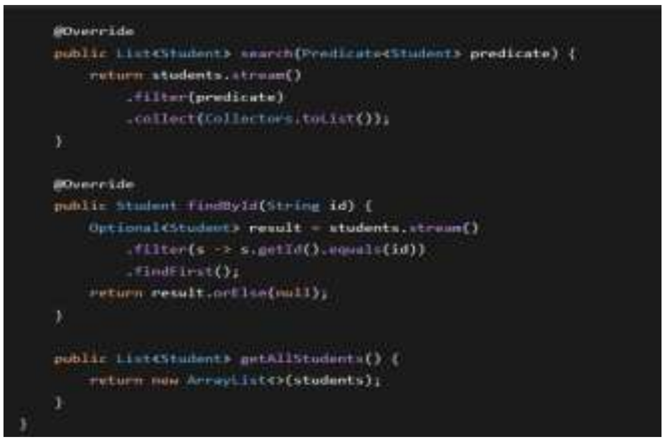
src/edu/ccrm/service/StudentService.java



```java
package edu.ccrm.service;

import edu.ccrm.domain.Student;
import edu.ccrm.domain.Course;
import edu.ccrm.domain.Semester;
import edu.ccrm.exception.DuplicateEnrollmentException;
import edu.ccrm.exception.MaxCreditLimitExceededException;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.function.Predicate;
import java.util.stream.Collectors;

public class StudentService implements Searchable<Student> {
    private List<Student> students;
    private static final int MAX_CREDITS_PER_SEMESTER = 18;

    public StudentService() {
        this.students = new ArrayList<>();
    }
```



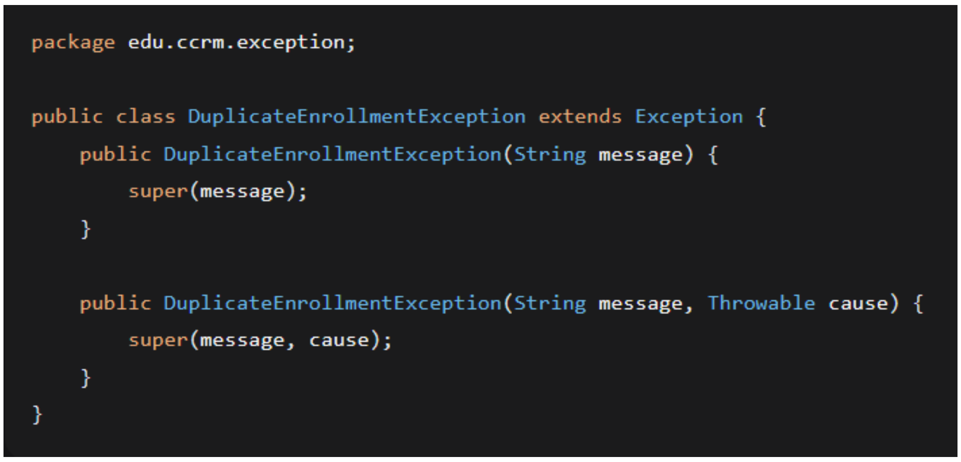
```java
    public void addStudent(Student student) {
        students.add(student);
    }

    public void enrollStudentInCourse(Student student, Course course, Semester semeste
r)
            throws DuplicateEnrollmentException, MaxCreditLimitExceededException {

        // Check for duplicate enrollment
        boolean alreadyEnrolled = student.getEnrollments().stream()
            .anyMatch(e -> e.getCourse().equals(course) && e.getSemester() == semeste
r);

        if (alreadyEnrolled) {
            throw new DuplicateEnrollmentException(
                "Student " + student.getRegNo() + " is already enrolled in " + course.
etCode());
        }

        // Check credit limit
        int currentCredits = student.getEnrollments().stream()
```



```java
    @Override
    public List<Student> search(Predicate<Student> predicate) {
        return students.stream()
            .filter(predicate)
            .collect(Collectors.toList());
    }

    @Override
    public Student findById(String id) {
        Optional<Student> result = students.stream()
            .filter(s -> s.getId().equals(id))
            .findFirst();
        return result.orElse(null);
    }

    public List<Student> getAllStudents() {
        return new ArrayList<>(students);
    }
}
```

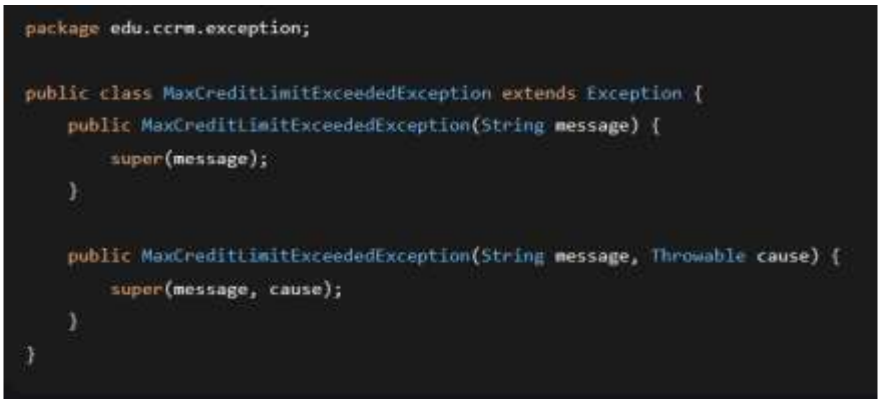
7. Exception Classes

src/edu/ccrm/exception/DuplicateEnrollmentException.java



```java
package edu.ccrm.exception;

public class DuplicateEnrollmentException extends Exception {
    public DuplicateEnrollmentException(String message) {
        super(message);
    }

    public DuplicateEnrollmentException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

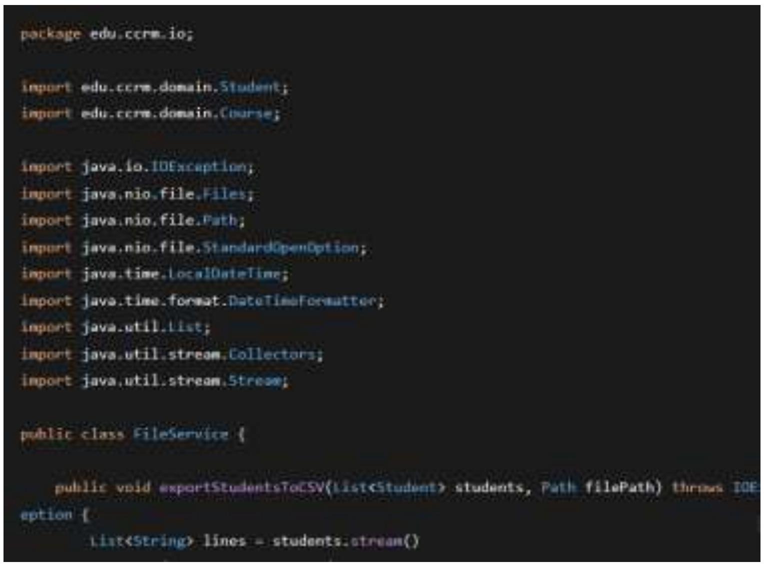
src/edu/ccrm/exception/MaxCreditLimitExceededException.java



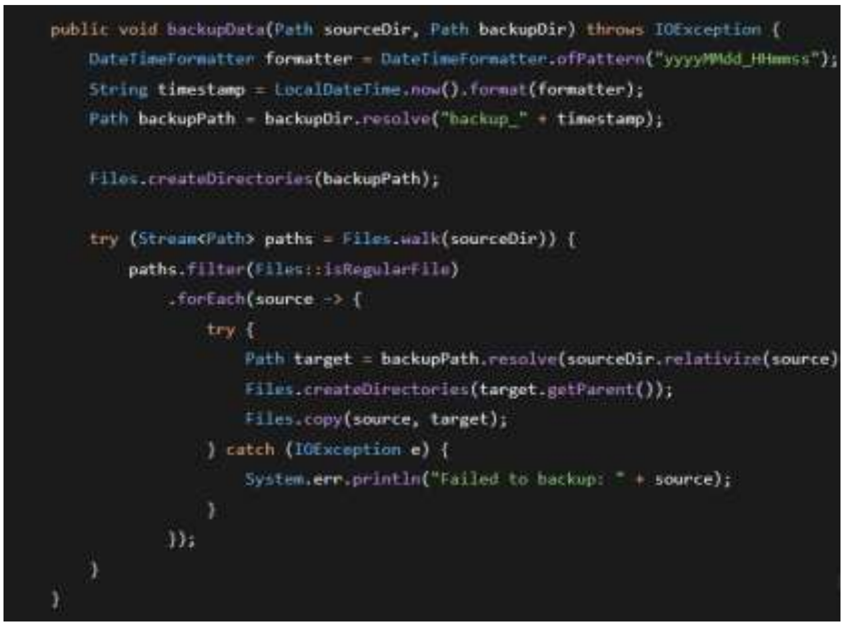
```java
package edu.ccrm.exception;

public class MaxCreditLimitExceededException extends Exception {
    public MaxCreditLimitExceededException(String message) {
        super(message);
    }

    public MaxCreditLimitExceededException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

8. File I/O with NIO.2

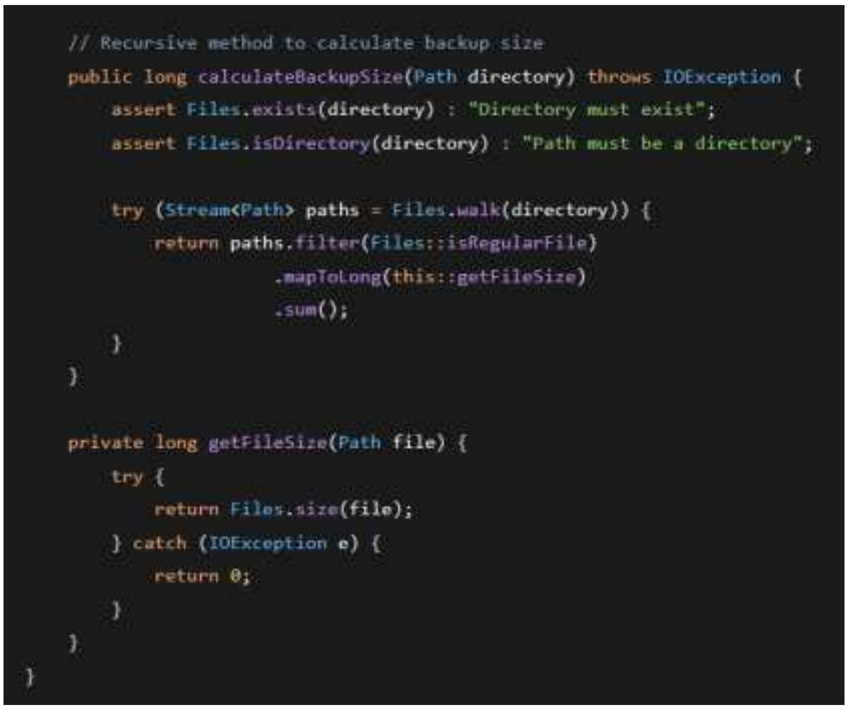
src/edu/ccrm/io/FileService.java



```java
package edu.ccrm.io;

import edu.ccrm.domain.Student;
import edu.ccrm.domain.Course;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardOpenOption;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class FileService {

    public void exportStudentsToCSV(List<Student> students, Path filePath) throws IOE
eption {
        List<String> lines = students.stream()
```



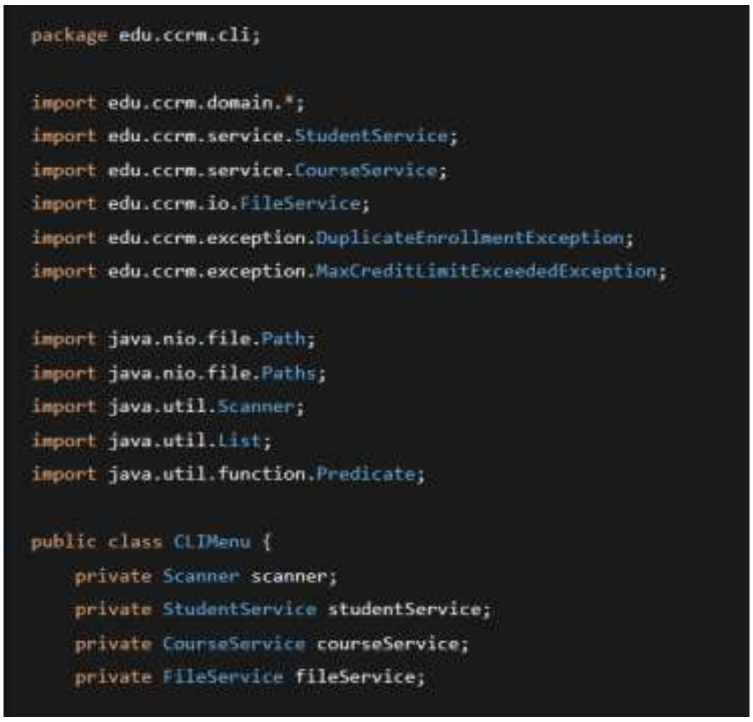
```java
    public void backupData(Path sourceDir, Path backupDir) throws IOException {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd_HHmmss");
        String timestamp = LocalDateTime.now().format(formatter);
        Path backupPath = backupDir.resolve("backup_" + timestamp);

        Files.createDirectories(backupPath);

        try (Stream<Path> paths = Files.walk(sourceDir)) {
            paths.filter(Files::isRegularFile)
                .forEach(source -> {
                    try {
                        Path target = backupPath.resolve(sourceDir.relativize(source)
                        Files.createDirectories(target.getParent());
                        Files.copy(source, target);
                    } catch (IOException e) {
                        System.err.println("Failed to backup: " + source);
                    }
                });
        }
    }
```



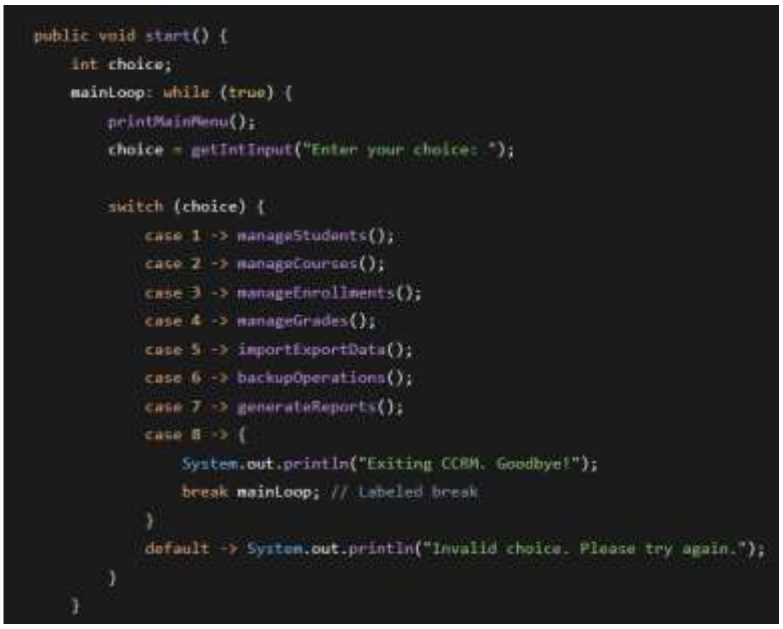
```java
    // Recursive method to calculate backup size
    public long calculateBackupSize(Path directory) throws IOException {
        assert Files.exists(directory) : "Directory must exist";
        assert Files.isDirectory(directory) : "Path must be a directory";

        try (Stream<Path> paths = Files.walk(directory)) {
            return paths.filter(Files::isRegularFile)
                        .mapToLong(this::getFileSize)
                        .sum();
        }
    }

    private long getFileSize(Path file) {
        try {
            return Files.size(file);
        } catch (IOException e) {
            return 0;
        }
    }
}
```

9. CLI Menu System

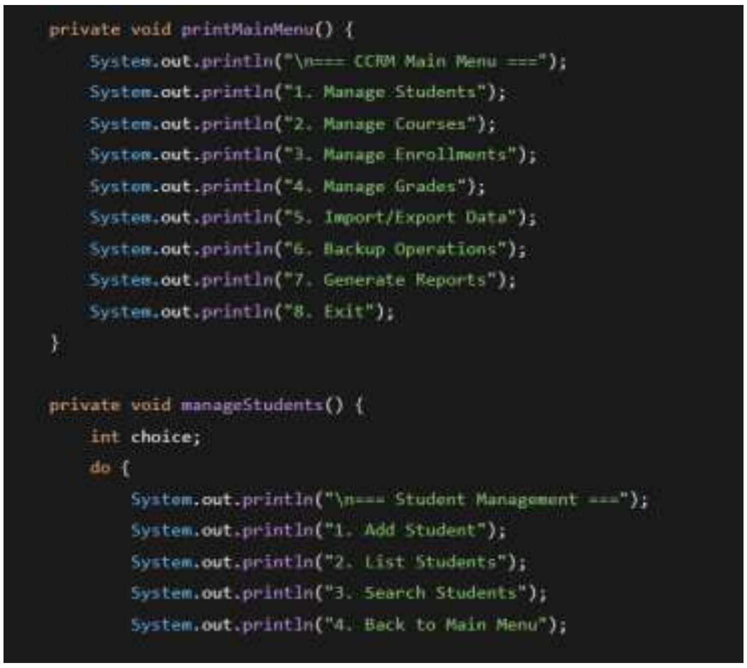
src/edu/ccrm/cli/CLIMenu.java



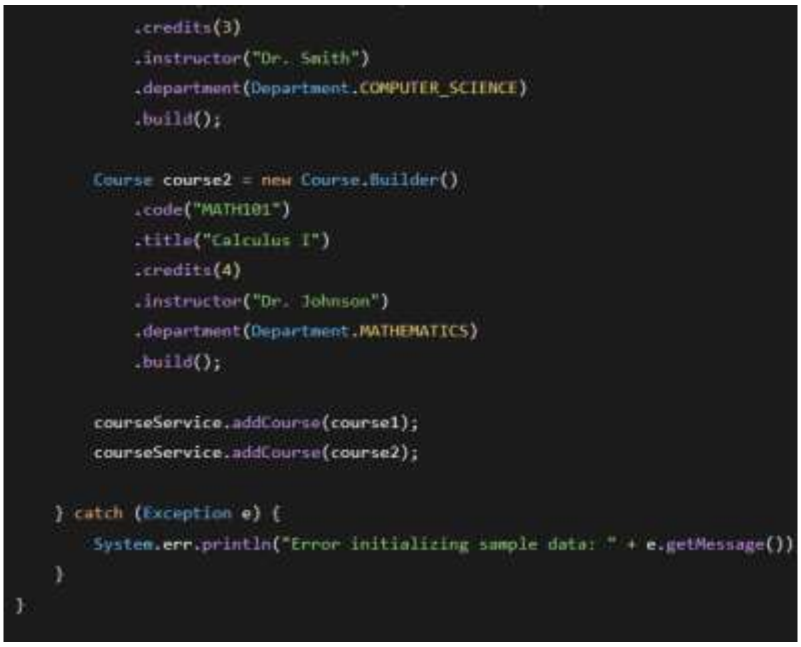
```java
package edu.ccrm.cli;

import edu.ccrm.domain.*;
import edu.ccrm.service.StudentService;
import edu.ccrm.service.CourseService;
import edu.ccrm.io.FileService;
import edu.ccrm.exception.DuplicateEnrollmentException;
import edu.ccrm.exception.MaxCreditLimitExceededException;

import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Scanner;
import java.util.List;
import java.util.function.Predicate;

public class CLIMenu {
    private Scanner scanner;
    private StudentService studentService;
    private CourseService courseService;
    private FileService fileService;
```



```java
public void start() {
    int choice;
    mainLoop: while (true) {
        printMainMenu();
        choice = getIntInput("Enter your choice: ");

        switch (choice) {
            case 1 -> manageStudents();
            case 2 -> manageCourses();
            case 3 -> manageEnrollments();
            case 4 -> manageGrades();
            case 5 -> importExportData();
            case 6 -> backupOperations();
            case 7 -> generateReports();
            case 8 -> {
                System.out.println("Exiting CCRM. Goodbye!");
                break mainLoop; // Labeled break
            }
            default -> System.out.println("Invalid choice. Please try again.");
        }
    }
```



```java
    private void printMainMenu() {
        System.out.println("\n=== CCRM Main Menu ===");
        System.out.println("1. Manage Students");
        System.out.println("2. Manage Courses");
        System.out.println("3. Manage Enrollments");
        System.out.println("4. Manage Grades");
        System.out.println("5. Import/Export Data");
        System.out.println("6. Backup Operations");
        System.out.println("7. Generate Reports");
        System.out.println("8. Exit");
    }

    private void manageStudents() {
        int choice;
        do {
            System.out.println("\n=== Student Management ===");
            System.out.println("1. Add Student");
            System.out.println("2. List Students");
            System.out.println("3. Search Students");
            System.out.println("4. Back to Main Menu");
```



```java
                .credits(3)
                .instructor("Dr. Smith")
                .department(Department.COMPUTER_SCIENCE)
                .build();

            Course course2 = new Course.Builder()
                .code("MATH101")
                .title("Calculus I")
                .credits(4)
                .instructor("Dr. Johnson")
                .department(Department.MATHEMATICS)
                .build();

            courseService.addCourse(course1);
            courseService.addCourse(course2);

        } catch (Exception e) {
            System.err.println("Error initializing sample data: " + e.getMessage())
        }
}
```