**CSCI 428 – Spring 2024**

**Ashish Acharya**

**Final Project**

For this project, I've created an object-oriented Java application for managing student grade reports. The program incorporates encapsulation, inheritance, composition, and polymorphism to enhance functionality and maintainability. Encapsulation is used to encapsulate data and behavior within classes. Inheritance is seen in the relationship between the "Student" and "Person" classes. Composition is employed in the "GradeReport" class, which contains a list of "SubjectScore" objects. Polymorphism is demonstrated in the "SubjectScore" class, allowing for different grading criteria. Overall, these principles contribute to an efficient and adaptable system for managing student grades.

**UML CLASS DIAGRAM:**

The UML diagram outlines a Java application for managing student grade reports. It includes classes for persons and students, with associated attributes like name and age. The "GradeReport" class connects students with subject scores, enabling report generation and grade calculation. "SubjectScore" objects hold subject names and numerical scores, with methods for determining letter grades. A UI controller, "HelloController," handles input elements and calculations, while "HelloApplication" serves as the main class for initiating the JavaFX interface.

A screenshot of a computer program

Description automatically generated

**SOURCE CODE:**

Source code of my project is separated into different files so that it can be analyzed, maintained, and integrated in an easy way to maintain good readability.

**Main.java**

***/\*\*  
 Program prompts the user to enter the name, Age and score for 4 subjects  
 calculate letter grades for each subject  
 calculate final grades  
 @assignment CSCI 428 Project  
 @date 04/28/2024  
 Name: Ashish Acharya  
 CWID: 50337433  
 \*\*/*package project.project;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
 @Override  
 public void start(Stage stage) throws Exception {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("hello-view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 600, 700);  
 stage.setTitle("Student Grading System");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}**

**MainController.java**

***/\*\*  
 Program prompts the user to enter the name, Age and score for 4 subjects  
 calculate letter grades for each subject  
 calculate final grades  
 @assignment CSCI 428 Project  
 @date 04/28/2024  
 Name: Ashish Acharya  
 CWID: 50337433  
 \*\*/*package project.project;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class MainController {  
 @FXML private TextField studentNameField;  
 @FXML private TextField studentAgeField;  
 @FXML private Label resultLabel;  
  
 *// Fields for 4 subjects* @FXML private TextField scoreField1, scoreField2, scoreField3, scoreField4;  
  
 @FXML  
 protected void onCalculateButtonClick() {  
 try {  
 String name = studentNameField.getText();  
 int age = Integer.*parseInt*(studentAgeField.getText());  
  
 List<SubjectScore> scores = new ArrayList<>();  
 scores.add(new SubjectScore("Subject 1", Integer.*parseInt*(scoreField1.getText())));  
 scores.add(new SubjectScore("Subject 2", Integer.*parseInt*(scoreField2.getText())));  
 scores.add(new SubjectScore("Subject 3", Integer.*parseInt*(scoreField3.getText())));  
 scores.add(new SubjectScore("Subject 4", Integer.*parseInt*(scoreField4.getText())));  
  
 Student student = new Student(name, age);  
 GradeReport gradeReport = new GradeReport(student, scores);  
  
 resultLabel.setText(gradeReport.generateReport());  
 } catch (NumberFormatException e) {  
 resultLabel.setText("Please enter valid numbers for age and scores.");  
 } catch (Exception e) {  
 resultLabel.setText("An error occurred: " + e.getMessage());  
 }  
 }  
}**

**Person.java (Encapsulation and Inheritance)**

Person.java class showcases encapsulation by keeping the `name` property private and providing a public method to access it, ensuring controlled interaction with the class’s data. It also sets up inheritance as an abstract base class for other classes to extend, allowing them to inherit common properties and behaviors, such as the `name` property and any methods defined in `Person`.

**package project.project;  
  
*// Abstract class representing a generic person in the system*public abstract class Person {  
 private String name;  
  
 *// Constructor to initialize the Person object with a name* public Person(String name) {  
 this.name = name;  
 }  
  
 *// Getter method to access the private name field* public String getName() {  
 return name;  
 }  
}**

**Student.java(Inheritance)**

Student class exemplifies inheritance by deriving from the `Person` class, thereby inheriting its attributes and functionalities, while also introducing additional characteristics unique to students, such as `age`. This approach enhances code reusability and establishes a clear hierarchical structure within the object model.

**package project.project;  
  
*//Extends Person to include student-specific attributes.*public class Student extends Person {  
 private int age; *// Student's age  
  
 //Constructs a student with name and age.* public Student(String name, int age) {  
 super(name); *// Delegate name initialization to the superclass* this.age = age;  
 }  
  
   
 public int getAge() {  
 return age; *// return student age* }  
}**

**SubjectScore.java(Encapsulation)**

SubjectScore class demonstrates encapsulation by keeping the `subjectName` and `score` properties private and providing public methods to access and modify these properties, ensuring controlled and safe interaction with the class's data.

**package project.project;  
  
public class SubjectScore {  
 private String subjectName;  
 private int score;  
 *// Constructor to initialize subjectName and score* public SubjectScore(String subjectName, int score) {  
 this.subjectName = subjectName;  
 this.score = score;  
 }  
 *// Getter method to retrieve subjectName* public String getSubjectName() {  
 return subjectName;  
 }  
 *// Getter method to retrieve score* public int getScore() {  
 return score;  
 }  
  
 *// Method to calculate and return the letter grade based on the score* public String getLetterGrade() {  
 *// Calculate letter grade based on score* if (score >= 90)  
 return "A"; *// 'A' for scores 90 or above* else if (score >= 80)  
 return "B"; *// 'B' for scores between 80 and 89* else if (score >= 70)  
 return "C"; *// 'C' for scores between 70 and 79* else if (score >= 60)  
 return "D"; *// 'D' for scores between 60 and 69* else  
 return "F"; *// 'F' for scores below 60* }  
}**

**GradeReport.java(Composition )**

GradeReport class uses composition by incorporating a `Student` and `SubjectScore` objects, forming a complete student grade report.

**package project.project;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class GradeReport {  
 *// Fields for storing student and their subject scores* private Student student;  
 private List<SubjectScore> subjectScores;  
  
 *// Constructor to initialize a GradeReport object with a student and their scores* public GradeReport(Student student, List<SubjectScore> subjectScores) {  
 this.student = student;  
 this.subjectScores = subjectScores;  
 }  
  
 *// Getter for student* public Student getStudent() {  
 return student;  
 }  
  
 *// Getter for subject scores* public List<SubjectScore> getSubjectScores() {  
 return subjectScores;  
 }  
  
 *// Generates a textual grade report for the student* public String generateReport() {  
 String report = "Grade Report for " + student.getName() + ":\n";  
 report += subjectScores.stream()  
 .map(score -> score.getSubjectName() + ": " + score.getLetterGrade())  
 .collect(Collectors.*joining*("\n")) + "\n";  
  
 *// Append the final grade to the report* report += "Final Grade: " + getFinalLetterGrade();  
 return report;  
 }  
  
 *// Calculates the average score from all subject scores* public double calculateAverageScore() {  
 return subjectScores.stream()  
 .mapToInt(SubjectScore::getScore)  
 .average()  
 .orElse(0.0); *// Provide a default value if no scores exist* }  
  
 *// Returns the final letter grade based on the average score* public String getFinalLetterGrade() {  
 return new SubjectScore("Final Grade", (int)calculateAverageScore()).getLetterGrade();  
 }  
}**

**Output:**

The output shows the grade for each subject and final grade overall.

**A screenshot of a computer

Description automatically generated**