COMSATS University Islamabad, Abbottabad Campus

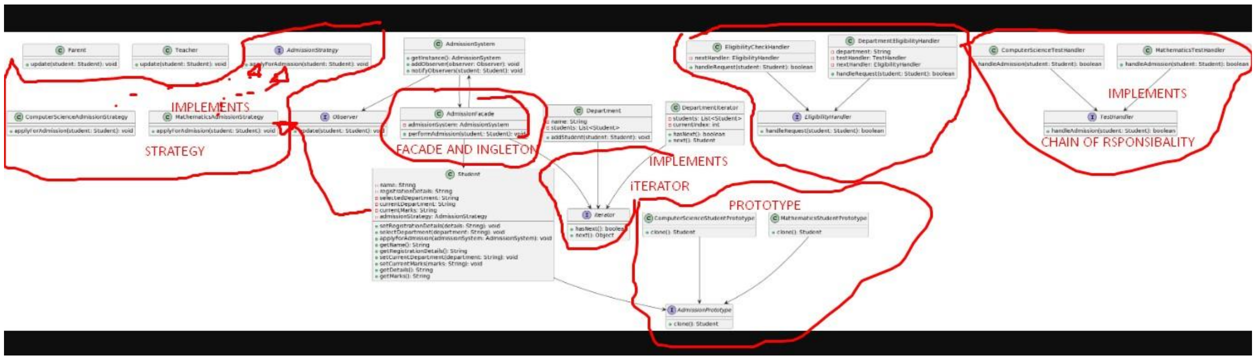
Department of Computer Science

Lab Final

Class: **BSE 7A** Date: 28 Dec 2023

Subject: Design Pattern Instructor: Mukhtiar Zamin

Name:**Quaid Ahmed** Registration #**FA20-BSE-034**



```java
package FinalExam;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

// Observer Design Pattern

// Subject
class AdmissionSystem {
    private List<Observer> observers = new ArrayList<>();

    private static AdmissionSystem instance;

    private AdmissionSystem() {
        // private constructor to enforce Singleton pattern
    }

    public static synchronized AdmissionSystem getInstance() {
        if (instance == null) {
            instance = new AdmissionSystem();
        }
        return instance;
    }
    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    public void notifyObservers(Student student) {
        for (Observer observer : observers) {
            observer.update(student);
        }
    }
}

// Observer
interface Observer {
    void update(Student student);
}

// Concrete Observers
class Parent implements Observer {
    public void update(Student student) {
        System.out.println("Parent notified about admission updates for " +
student.getName());
        System.out.println("Details: " + student.getDetails());
    }
}

class Teacher implements Observer {
    public void update(Student student) {
        System.out.println("Teacher notified about admission updates for " +
student.getName());
        System.out.println("Details: " + student.getDetails());
    }
}

// Strategy Design Pattern

// Strategy
interface AdmissionStrategy {
    void applyForAdmission(Student student);
}

// Concrete Strategies
class ComputerScienceAdmissionStrategy implements AdmissionStrategy {
    @Override
    public void applyForAdmission(Student student) {
        System.out.println(student.getName() + " has applied for Computer
Science admission.");
        student.setCurrentDepartment("Computer Science");
    }
}

class MathematicsAdmissionStrategy implements AdmissionStrategy {
    @Override
    public void applyForAdmission(Student student) {
        System.out.println(student.getName() + " has applied for Mathematics
admission.");
        student.setCurrentDepartment("Mathematics");
    }
```

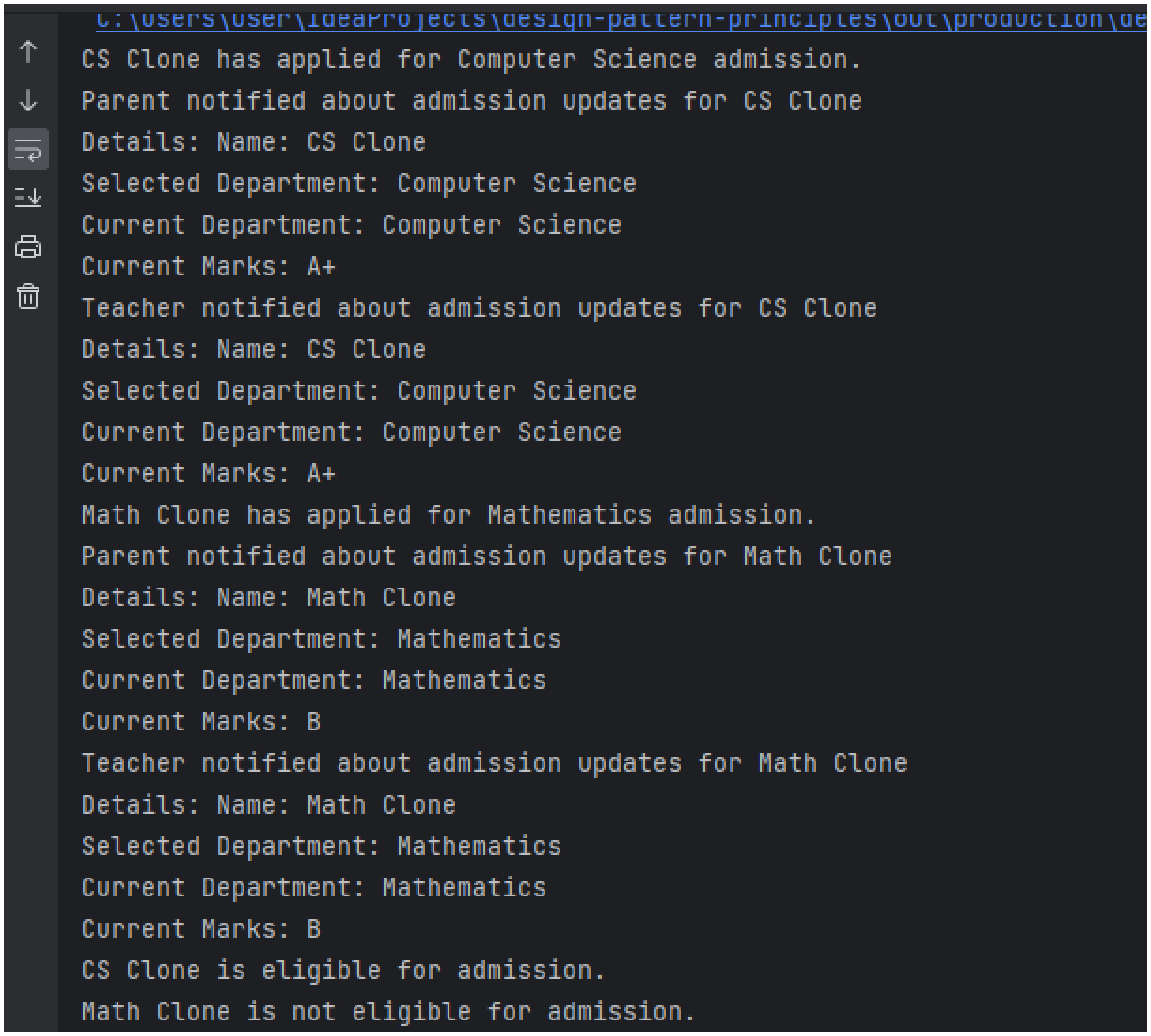
```java
}

// Context
class Student {
    private String name;
    private String registrationDetails;
    private String selectedDepartment;
    private String currentDepartment;
    private String currentMarks;
    private AdmissionStrategy admissionStrategy;

    public Student(String name, AdmissionStrategy admissionStrategy) {
        this.name = name;
        this.admissionStrategy = admissionStrategy;
    }

    public void setRegistrationDetails(String details) {
        this.registrationDetails = details;
    }

    public void selectDepartment(String department) {
        this.selectedDepartment = department;
    }

    public void applyForAdmission(AdmissionSystem admissionSystem) {
        admissionStrategy.applyForAdmission(this);
        admissionSystem.notifyObservers(this);
    }

    public String getName() {
        return name;
    }

    public String getRegistrationDetails() {
        return registrationDetails;
    }

    public void setCurrentDepartment(String department) {
        this.currentDepartment = department;
    }

    public void setCurrentMarks(String marks) {
        this.currentMarks = marks;
    }

    public String getDetails() {
        return "Name: " + name +
                "\nSelected Department: " + selectedDepartment +
                "\nCurrent Department: " + currentDepartment +
                "\nCurrent Marks: " + currentMarks;
    }

    public String getMarks() {
        return currentMarks;
    }
}
```

```java
// Facade Design Pattern

// Facade
class AdmissionFacade {
    private AdmissionSystem admissionSystem;

    public AdmissionFacade(AdmissionSystem admissionSystem) {
        this.admissionSystem = admissionSystem;
    }

    public void performAdmission(Student student) {
        student.applyForAdmission(admissionSystem);
    }
}

// Iterator Design Pattern

// Aggregate
class Department implements Iterable<Student> {
    private String name;
    private List<Student> students = new ArrayList<>();

    public Department(String name) {
        this.name = name;
    }

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

    @Override
    public Iterator<Student> iterator() {
        return new DepartmentIterator(students);
    }
}

// Concrete Iterator
class DepartmentIterator implements Iterator<Student> {
    private List<Student> students;
    private int currentIndex = 0;

    public DepartmentIterator(List<Student> students) {
        this.students = students;
    }

    public boolean hasNext() {
        return currentIndex < students.size();
    }

    public Student next() {
        return students.get(currentIndex++);
    }
}

// Chain of Responsibility Design Pattern

// Handler
```

```java
interface EligibilityHandler {
    boolean handleRequest(Student student);
}

// Concrete Handlers
class EligibilityCheckHandler implements EligibilityHandler {
    private EligibilityHandler nextHandler;

    public EligibilityCheckHandler(EligibilityHandler nextHandler) {
        this.nextHandler = nextHandler;
    }

    @Override
    public boolean handleRequest(Student student) {
        // Perform general eligibility check logic (dummy logic)
        boolean isEligible =
student.getRegistrationDetails().contains("Excellent");
        if (!isEligible && nextHandler != null) {
            return nextHandler.handleRequest(student);
        }
        return isEligible;
    }
}

class DepartmentEligibilityHandler implements EligibilityHandler {
    private String department;
    private TestHandler testHandler;
    private EligibilityHandler nextHandler;

    public DepartmentEligibilityHandler(String department, TestHandler
testHandler, EligibilityHandler nextHandler) {
        this.department = department;
        this.testHandler = testHandler;
        this.nextHandler = nextHandler;
    }

    @Override
    public boolean handleRequest(Student student) {
        // Perform department-specific eligibility check logic using the test
handler
        return testHandler.handleAdmission(student) || (nextHandler != null
&& nextHandler.handleRequest(student));
    }
}

// TestHandler Interface
interface TestHandler {
    boolean handleAdmission(Student student);
}

// Concrete TestHandlers
class ComputerScienceTestHandler implements TestHandler {
    @Override
    public boolean handleAdmission(Student student) {
        // Perform Computer Science department admission test logic (dummy
logic)
        return student.getMarks().startsWith("A"); // Example: Students with
```

```java
marks starting with 'A' are eligible
    }
}

class MathematicsTestHandler implements TestHandler {
    @Override
    public boolean handleAdmission(Student student) {
        // Perform Mathematics department admission test logic (dummy logic)
        return student.getMarks().contains("A+"); // Example: Students with
A+ marks are eligible
    }
}

// Prototype Design Pattern

// Prototype
interface AdmissionPrototype {
    Student clone();
}

// Concrete Prototypes
class ComputerScienceStudentPrototype implements AdmissionPrototype {
    @Override
    public Student clone() {
        return new Student("CS Clone", new
ComputerScienceAdmissionStrategy());
    }
}


class MathematicsStudentPrototype implements AdmissionPrototype {
    @Override
    public Student clone() {
        return new Student("Math Clone", new MathematicsAdmissionStrategy());
    }
}

// Main Application
public class UniversityAdmissionSystem {
    public static void main(String[] args) {
        // Singleton pattern: Get instance of the AdmissionSystem
        AdmissionSystem admissionSystem = AdmissionSystem.getInstance();

        // Observer pattern: Add observers
        Observer parent = new Parent();
        Observer teacher = new Teacher();
        admissionSystem.addObserver(parent);
        admissionSystem.addObserver(teacher);

        // Facade pattern: Create admission facade
        AdmissionFacade admissionFacade = new
AdmissionFacade(admissionSystem);

        // Iterator pattern: Create departments with test handlers
        TestHandler computerScienceTestHandler = new
ComputerScienceTestHandler();
        Department computerScience = new Department("Computer Science");
        TestHandler mathematicsTestHandler = new MathematicsTestHandler();
        Department mathematics = new Department("Mathematics");

        // Chain of Responsibility pattern: Create eligibility handlers
        EligibilityHandler generalEligibilityHandler = new
EligibilityCheckHandler(null);
        EligibilityHandler csEligibilityHandler = new
DepartmentEligibilityHandler("Computer Science", computerScienceTestHandler,
generalEligibilityHandler);
        EligibilityHandler mathematicsEligibilityHandler = new
DepartmentEligibilityHandler("Mathematics", mathematicsTestHandler,
generalEligibilityHandler);

        // Prototype pattern: Create student prototypes
        AdmissionPrototype computerScienceStudentPrototype = new
ComputerScienceStudentPrototype();
        AdmissionPrototype mathematicsStudentPrototype = new
MathematicsStudentPrototype();

        // Create students with admission strategies using prototypes
        Student student1 = computerScienceStudentPrototype.clone();
        Student student2 = mathematicsStudentPrototype.clone();

        // Students add details and apply for admission using the facade
        student1.setRegistrationDetails("High School Transcript: Excellent");
        student1.selectDepartment("Computer Science");
        student1.setCurrentMarks("A+");
        admissionFacade.performAdmission(student1);

        student2.setRegistrationDetails("High School Transcript: Good");
        student2.selectDepartment("Mathematics");
        student2.setCurrentMarks("B");
        admissionFacade.performAdmission(student2);

        // Chain of Responsibility pattern: Perform eligibility checks
        if (generalEligibilityHandler.handleRequest(student1)) {
            System.out.println(student1.getName() + " is eligible for
admission.");
        } else {
            System.out.println(student1.getName() + " is not eligible for
admission.");
        }

        if (generalEligibilityHandler.handleRequest(student2)) {
            System.out.println(student2.getName() + " is eligible for
admission.");
        } else {
            System.out.println(student2.getName() + " is not eligible for
admission.");
        }

        // Iterator pattern: Iterate over the results of the admission tests
        computerScience.addStudent(student1);
        computerScience.addStudent(student2);

        System.out.println("\nResults of Computer Science Department
Admission Tests:");
        Iterator<Student> csIterator = computerScience.iterator();
        while (csIterator.hasNext()) {
            Student csStudent = csIterator.next();
            System.out.println(csStudent.getDetails());
        }

        mathematics.addStudent(student2);

        System.out.println("\nResults of Mathematics Department Admission
Tests:");
        Iterator<Student> mathIterator = mathematics.iterator();
        while (mathIterator.hasNext()) {
            Student mathStudent = mathIterator.next();
            System.out.println(mathStudent.getDetails());
        }
    }

}
```

output



```
C:\Users\user\IdeaProjects\design-pattern-principles\out\production\de
CS Clone has applied for Computer Science admission.
Parent notified about admission updates for CS Clone
Details: Name: CS Clone
Selected Department: Computer Science
Current Department: Computer Science
Current Marks: A+
Teacher notified about admission updates for CS Clone
Details: Name: CS Clone
Selected Department: Computer Science
Current Department: Computer Science
Current Marks: A+
Math Clone has applied for Mathematics admission.
Parent notified about admission updates for Math Clone
Details: Name: Math Clone
Selected Department: Mathematics
Current Department: Mathematics
Current Marks: B
Teacher notified about admission updates for Math Clone
Details: Name: Math Clone
Selected Department: Mathematics
Current Department: Mathematics
Current Marks: B
CS Clone is eligible for admission.
Math Clone is not eligible for admission.
```