|  |  |
| --- | --- |
| Course | Advanced Software Design – CS525 |
| Assignment | Lab 6 |
| Week | 06 |
| Due | 8/03/2025 |
| Student name | Toe Toe Aung |
| Student ID | 618090 |

1. **Class Diagram of Game Application**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Sequence Diagram of Game Application**

**A diagram of a computer program

AI-generated content may be incorrect.**

1. **Implementation of Game Application in Java**

package Game;

public class Game {

private LevelState state;

public void play() {

this.state.play();

}

public void changeLevelState(LevelState state) {

this.state = state;

}

public void addPoints(int points) {

this.state.setPoints(points);

}

}

public class Level1State implements LevelState{

private int points = 0;

private Game g;

public Level1State(Game g) {

this.g = g;

}

@Override

public void setPoints(int points) {

this.points += points;

}

@Override

public void play() {

if (this.points > 10) {

g.changeLevelState(new Level2State(g));

g.addPoints(points);

g.play();

}

else {

System.*out*.println("Total points = " + points + " --- level = 1");

}

}

}

public class Level2State implements LevelState {

private Game g;

private int points = 0;

public Level2State(Game g) {

this.g = g;

}

@Override

public void setPoints(int points) {

this.points += points;

}

@Override

public void play() {

if (this.points > 10 && this.points <= 15) {

System.*out*.println("Total points = " + points + " --- level = 2");

} else {

g.changeLevelState(new Level2\_5State(g));

g.addPoints(points);

g.play();

}

}

}

public class Level3State implements LevelState {

private int points = 0;

private Game g;

public Level3State(Game g) {

this.g = g;

}

@Override

public void setPoints(int points) {

this.points += points;

}

@Override

public void play() {

System.*out*.println("Total points = " + points + " --- level = 3");

}

}

public interface LevelState {

void setPoints(int points);

void play();

}

1. **The observer pattern of lab 3, the command pattern of lab 5 and the state pattern in one diagram**

**A diagram of a computer

AI-generated content may be incorrect.**

1. **The Sequence Diagram that shows the following scenario:**

**a. The user clicks the increment button**

**b. The user clicks undo**

**A diagram of a diagram

AI-generated content may be incorrect.**

1. **Implementation of the new design in Java using the observer and command pattern.**

**package Counter;**

**public interface ICounterState {**

**void update();**

**void setFlag(boolean f);**

**void setPoints(int points);**

**}**

**public class IncrementCommand implements ICounterCommand {**

**private Counter counter;**

**public IncrementCommand(Counter counter) {**

**this.counter = counter;**

**}**

**@Override**

**public void execute() {**

**counter.increment();**

**}**

**}**

**package Counter;**

**public class SingleDigitState implements ICounterState {**

**Counter c;**

**private int points = 0;**

**private boolean incDecFlag = true;**

**public SingleDigitState(Counter c, int points, boolean f) {**

**this.c = c;**

**}**

**@Override**

**public void update() {**

**if (incDecFlag) {**

**this.points++;**

**} else {**

**this.points--;**

**}**

**System.out.println("SingleState updated points: " + this.points);**

**if (this.points <= 0) {**

**this.points = 0;**

**} else if (this.points >= 10) {**

**c.changeState(new DoubleDigitState(c, this.points, incDecFlag));**

**}**

**c.setCount(this.points);**

**}**

**@Override**

**public void setFlag(boolean f) {**

**this.incDecFlag = f;**

**}**

**@Override**

**public void setPoints(int points) {**

**this.points = points;**

**}**

**}**