Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет (институт) Информационных технологий

Кафедра Прикладная математик

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М. Старолетов

(подпись преподавателя) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

Отчет

по лабораторной (практической) работе № 7

по дисциплине Архитектурное проектирование и паттерны программирования

(наименование дисциплины)

ЛР 09.03.04.10.000 ОТ

(обозначение документа)

Студенты группы ПИ-42 Е.А. Колесников

(инициалы, фамилии)

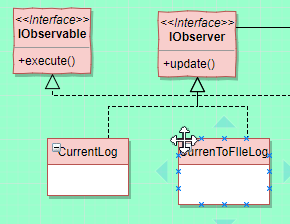
Преподаватель Доцент к. канд. физ.-мат. наук С.М.Старолетов

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2017

**Задание:** реализовать паттерны: Наблюдатель, Состояние, Хранитель

Наблюдатель:

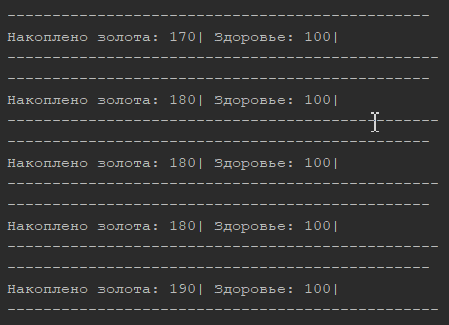


public interface IObserver {  
 public void update (int money,int hp) throws IOException;  
}

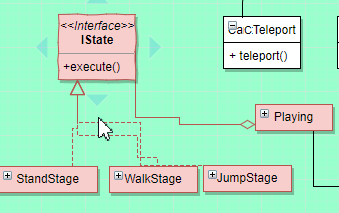
public interface IObservable {  
 void registerObserver(IObserver o);  
 void removeObserver(IObserver o);  
 void notifyObservers();  
}

public class CurrentToFileLog implements IObserver {  
 private Person person;  
 private OutputStream os = null;  
 public CurrentToFileLog(Person person) {  
 this.person = person;  
 person.registerObserver(this);  
 try {  
 os = new FileOutputStream(new File("log.txt"));  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void update(int money, int hp) throws IOException {  
  
 String s="-----------------------------------------------\n" +  
 "Накоплено золота: "+ money+"| Здоровье: "+hp+"|\n" +  
 "------------------------------------------------";  
 try {  
 os.write(s.getBytes());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

public class CurrentToFileLog implements IObserver {  
 private Person person;  
 private OutputStream os = null;  
 public CurrentToFileLog(Person person) {  
 this.person = person;  
 person.registerObserver(this);  
 try {  
 os = new FileOutputStream(new File("log.txt"));  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void update(int money, int hp) throws IOException {  
  
 String s="-----------------------------------------------\n" +  
 "Накоплено золота: "+ money+"| Здоровье: "+hp+"|\n" +  
 "------------------------------------------------";  
 try {  
 os.write(s.getBytes());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}



Cостояние:



public interface IState {  
 public void execute(Playing play, Person person);  
}

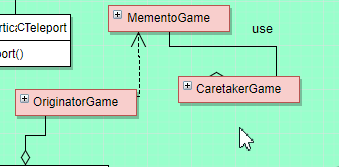
public class JumpState implements IState {  
 @Override  
 public void execute(Playing play, Person person) {  
 person.PerfomJump(TimeUtils.*nanoTime*());  
 if(!person.isOnGround())  
 play.setCurrentIState(new StandState());  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*LEFT*)) {  
 person.setOrientation(*ORIENTATION\_LEFT*);  
 play.setCurrentIState(new WalkState());  
 }  
 else  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*RIGHT*)) {  
 person.setOrientation(*ORIENTATION\_RIGHT*);  
 play.setCurrentIState(new WalkState());  
 }  
 else  
 play.setCurrentIState(new StandState());  
 }  
}

public class Playing {  
 private IState currentIState;  
  
 public Playing() {  
 this.currentIState = new StandState();  
 }  
  
 public void setCurrentIState(IState currentIState) {  
 this.currentIState = currentIState;  
 }  
  
 public void execute(Person person)  
 {  
 currentIState.execute(this,person);  
 }  
}

public class StandState implements IState {  
 @Override  
 public void execute(Playing play, Person person) {  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*LEFT*)) {  
 person.setOrientation(*ORIENTATION\_LEFT*);  
 play.setCurrentIState(new WalkState());  
 }  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*RIGHT*)) {  
 person.setOrientation(*ORIENTATION\_RIGHT*);  
 play.setCurrentIState(new WalkState());  
 }  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*UP*))  
 {  
 play.setCurrentIState(new JumpState());  
 }  
 }  
}

public class WalkState implements IState {  
 @Override  
 public void execute(Playing play, Person person) {  
  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*LEFT*)) {  
 person.perfomWalk();  
 person.setOrientation(*ORIENTATION\_LEFT*);  
 play.setCurrentIState(new WalkState());  
 }  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*RIGHT*)) {  
 person.perfomWalk();  
 person.setOrientation(*ORIENTATION\_RIGHT*);  
 play.setCurrentIState(new WalkState());  
 }  
 if (Gdx.*input*.isKeyPressed(com.badlogic.gdx.Input.Keys.*UP*))  
 {  
 play.setCurrentIState(new JumpState());  
 }  
 }  
}

Хранитель:



public class OriginatorGame {  
 private Games state;  
  
 public void setState(Games state) {  
 this.state = state;  
 }  
  
 public Games getState() {  
 return state;  
 }  
 public MementoGame saveState() {  
 return new MementoGame(state);  
 }  
 public void restoreState(MementoGame memento) {  
 this.state = memento.getState();  
 }  
}

public class MementoGame {  
 private final Games state;  
  
 public Games getState() {  
 return state;  
 }  
  
 public MementoGame(Games games) {  
  
 this.state = games;  
 }  
}

public class CaretakerGame {  
 private MementoGame mementoGame;  
  
 public MementoGame getMementoGame() {  
 return mementoGame;  
 }  
  
 public void setMementoGame(MementoGame mementoGame) {  
 this.mementoGame = mementoGame;  
 }  
}

