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 № 8

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)

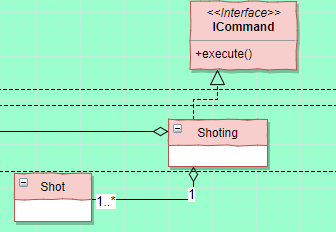
Преподаватель Доцент к. канд. физ.-мат. наук С.М.Старолетов

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2017

**Задание:** реализовать паттерны: Команда, перенаправление, посетитель

Команда:



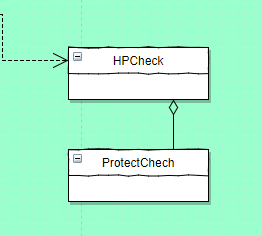
public interface ICommand {  
 public void execute();  
 public ArrayList<Shot> getShots();  
}

public class Shooting implements ICommand {  
 protected Person person;  
 private ArrayList<Shot> shots = new ArrayList<Shot>();  
  
 public Shooting(Person person) {  
 this.person = person;  
 this.person.saveCommand(this);  
  
 }  
  
 @Override  
 public ArrayList<Shot> getShots() {  
 return shots;  
 }  
  
 @Override  
 public void execute() {  
 shots.add(new Shot(person.getx(),person.getY()+32,person.getOrientation()));  
 }  
}

public Shot(float x, float y, int orient) {  
 this.orient = orient;  
 rectangle\_shot = new Rectangle();  
 rectangle\_shot.x = x;  
 rectangle\_shot.y = 52;  
 rectangle\_shot.height = 100;  
 rectangle\_shot.width = 100;  
 texture\_shot\_R = new Texture(Gdx.*files*.internal("super/fireballR.png"));  
 texture\_shot\_L = new Texture(Gdx.*files*.internal("super/fireballL.png"));  
 }  
  
 public Texture getShot(Person person) {  
 switch (orient) {  
 case 1:  
 return texture\_shot\_R;  
 case -1:  
 return texture\_shot\_L;  
 default:  
 return texture\_shot\_R;  
 }  
 }  
  
 public void update() {  
 rectangle\_shot.x += orient \* *VELOCITY\_X* \* Gdx.*graphics*.getDeltaTime();  
 }  
}

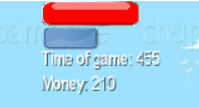


Перенаправление:

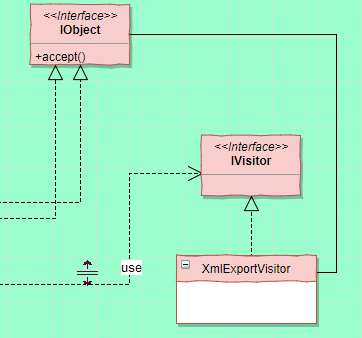


public class HPChek {  
 private ProtectChech protectChech=new ProtectChech();  
 public void ChangeHP(Person person, int damage)  
 {  
 int dam;  
 dam=protectChech.ChangeProtect(person,damage);  
 if(dam<0) {  
 person.setHP(person.getHP() + dam);  
 person.setProtect(0);  
 }  
 }  
}

public class ProtectChech {  
  
 public int ChangeProtect(Person person,int damage)  
 {  
 person.setProtect(person.getProtect()-damage);  
 return person.getProtect();  
 }  
}



Посетитель:



public interface IVisitingObject {  
 public String accept(IVisitor visitor);  
}

public interface IVisitor {  
 public String VisitPerson(Person person);  
 public String VisitFireLet(FireLet fireLet);  
 public String VisitRock(RockLet rockLet);  
}

public class XmlExportVisitor implements IVisitor {  
 private OutputStream os = null;  
  
 public XmlExportVisitor() {  
 try {  
 os = new FileOutputStream(new File("XmlParse.xml"));  
 String s="<?xml version=\"1.0\" encoding=\"utf-8\"?>" + "\n";  
 os.write(s.getBytes());  
 s="<Games>\n";  
 os.write(s.getBytes());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void export(IVisitingObject... args) {  
 StringBuilder sb = new StringBuilder();  
 for (IVisitingObject shape : args) {  
 sb.append(shape.accept(this) + "\n");  
  
 try {  
 os.write(sb.toString().getBytes());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 sb.setLength(0);  
 }  
 sb.append(this.VisitRock(new Rock()));  
 sb.append("</Games>");  
 try {  
 os.write(sb.toString().getBytes());  
 os.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 @Override  
 public String VisitPerson(Person person) {  
 return "<Person>" + "\n" +  
 " <X>" + person.getx() + "</X>\n" +  
 " <Y>" + person.getY() + "</Y>\n" +  
 " <Texteure\_Path>" + person.getPath\_player\_image() + "</Texteure\_Path>\n" +  
 " <HP>" + person.getHP() + "</HP>\n" +  
 " <Gold>" + person.getGold() + "</Gold>\n" +  
 "</Person>\n";  
 }  
  
 @Override  
 public String VisitFireLet(FireLet fireLet) {  
  
 return "<Fire\_Let>\n" +  
 " <X>" + fireLet.getHorizontal\_rectangle().x + "</X>\n" +  
 " <Y>" + fireLet.getHorizontal\_rectangle().x + "</Y>\n" +  
 "</Fire\_Let>\n";  
 }  
  
 @Override  
 public String VisitRock(RockLet rockLet) {  
 StringBuilder sb = new StringBuilder();  
 sb.append("<Poll\_Roklet>\n");  
 for (Map.Entry<RockLet, Boolean> entry : SinglePoolLet.*getInstance*().getPool().entrySet()) {  
 if (entry.getValue()) {  
 sb.append(" <" + entry.getKey().toString() + ">\n");  
 sb.append(" <X>" + entry.getKey().getRock\_let().x + "</X>\n");  
 sb.append(" <Y>" + entry.getKey().getRock\_let().x + "</Y>\n");  
 sb.append(" <Damage>" + entry.getKey().getDamage() + "</Damage>");  
 sb.append(" </" + entry.getKey().toString() + ">\n");  
 }  
 }  
 sb.append("</Poll\_Roklet>\n");  
 return sb.toString();  
 }  
}

