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 № 6

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)

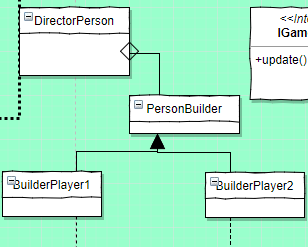
Преподаватель Доцент к. канд. физ.-мат. наук С.М.Старолетов

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2017

**Задание:** реализовать паттерны: Строитель, Сингтон, Пул

Строитель



public abstract class PersonBuilder {  
 protected Person player;  
  
 public Person getPlayer() {  
 return player;  
 }  
  
 public void createPlayer() {  
 player = new Person();  
 }  
  
 public abstract void buildPosition();  
  
 public abstract void buildVelocity();  
  
 public abstract void BuildTeleportAction();  
  
 public abstract void BuildWalkAction();  
  
 public abstract void BuildPlayerImage();  
  
 public abstract void BuildJumpAction();  
  
 public abstract void BuildRectangle();  
  
}

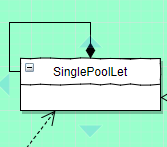
public class DirectorPerson {  
 private PersonBuilder personBuilder;  
  
 public void setPersonBuilder(PersonBuilder personBuilder) {  
 this.personBuilder = personBuilder;  
 }  
 public Person getPlayer()  
 {  
 return personBuilder.getPlayer();  
 }  
 public void buildPerson()  
 {  
 personBuilder.createPlayer();  
 personBuilder.BuildPlayerImage();  
 personBuilder.buildPosition();  
 personBuilder.buildVelocity();  
 personBuilder.BuildRectangle();  
 personBuilder.BuildTeleportAction();  
 personBuilder.BuildWalkAction();  
 }  
}

public class BuilderPlayer1 extends PersonBuilder {  
 @Override  
 public void buildPosition() {  
 player.setPosition(new Vector3(*VIEWPORT\_WIDTH* / 2 - *WIDTH\_PLAYER* / 2,*STARTER\_POSITION*,0));  
 }  
  
 @Override  
 public void buildVelocity() {  
 player.setVelocity(new Vector3(300, 600, 0));  
 }  
  
  
 @Override  
 public void BuildTeleportAction() {  
 player.setTeleportAction(new CanTeleport());  
 }  
  
 @Override  
 public void BuildWalkAction() {  
 player.setWalkAction(new CanWalk());  
 }  
  
 @Override  
 public void BuildPlayerImage() {  
 player.setPath\_player\_image("player");  
 player.setPlayerImage(new Texture(Gdx.*files*.internal("player/1.png")));  
  
 }  
  
 @Override  
 public void BuildJumpAction() {  
 player.setJumpAction(new CanJump());  
 }  
  
 @Override  
 public void BuildRectangle() {  
 Rectangle rectangle\_person = new Rectangle();  
 //центрируем персонажа по горизонтали  
 rectangle\_person.x = *VIEWPORT\_WIDTH* / 2 - *WIDTH\_PLAYER* / 2;  
 rectangle\_person.y = *STARTER\_POSITION*;  
 rectangle\_person.width = *WIDTH\_PLAYER*;  
 rectangle\_person.height = *HEIGHT\_PLAYER*;  
 player.setPlayer(rectangle\_person);  
 }  
}

public class BuilderPlayer2 extends PersonBuilder {  
 @Override  
 public void buildPosition() {  
 player.setPosition(new Vector3(*VIEWPORT\_WIDTH* / 2 - *WIDTH\_PLAYER* / 2, *STARTER\_POSITION*, 0));  
 }  
  
 @Override  
 public void buildVelocity() {  
 player.setVelocity(new Vector3(300, 600, 0));  
 }  
  
  
 @Override  
 public void BuildTeleportAction() {  
 player.setTeleportAction(new CanTeleport());  
 }  
  
 @Override  
 public void BuildWalkAction() {  
 player.setWalkAction(new CanWalk());  
 }  
  
 @Override  
 public void BuildJumpAction() {  
 player.setJumpAction(new CanJump());  
 }  
  
 @Override  
 public void BuildPlayerImage() {  
 player.setPath\_player\_image("player2");  
 player.setPlayerImage(new Texture(Gdx.*files*.internal("player2/1.png")));  
  
 }  
  
 @Override  
 public void BuildRectangle() {  
 Rectangle rectangle\_person = new Rectangle();  
 //центрируем персонажа по горизонтали  
 rectangle\_person.x = *VIEWPORT\_WIDTH* / 2 - *WIDTH\_PLAYER* / 2;  
 rectangle\_person.y = *STARTER\_POSITION*;  
 rectangle\_person.width = *WIDTH\_PLAYER*;  
 rectangle\_person.height = *HEIGHT\_PLAYER*;  
 player.setPlayer(rectangle\_person);  
 }  
}



Синглтон и пул:



public class SinglePoolLet {  
 public static SinglePoolLet *instance*;  
 private HashMap<RockLet,Boolean> pool;  
 private SinglePoolLet() {  
 pool=new HashMap<RockLet, Boolean>();  
 }  
 public static SinglePoolLet getInstance() {  
 if (*instance* == null)  
 *instance* = new SinglePoolLet();  
 return *instance*;  
 }  
 public Object createNewObject()  
 {  
 for(Map.Entry<RockLet,Boolean> entry : pool.entrySet())  
 {  
 if(!entry.getValue()) {  
 entry.setValue(new Boolean(true));  
 return entry;  
 }  
 }  
 RockLet rockLet=new Rock();  
 switch (MathUtils.*random*(0,1))  
 {  
 case 0: rockLet=new Rock();pool.put(rockLet,true);break;  
 case 1: rockLet=new IceRock();pool.put(rockLet,true);break;  
 }  
 return rockLet;  
 }  
 public void Delete(RockLet rockLet)  
 {  
 for(Map.Entry<RockLet,Boolean> entry : pool.entrySet())  
 {  
 if(entry.getKey()==rockLet) {  
 entry.setValue(new Boolean(false));  
 }  
 }  
 }  
 public void Move()  
 {  
 for(Map.Entry<RockLet,Boolean> entry : pool.entrySet()) {  
 if(entry.getValue())  
 entry.getKey().fall();  
 if(entry.getKey().getRock\_let().y<0) {  
 entry.setValue(new Boolean(false));  
 entry.getKey().create\_rock();  
 }  
 }  
 }  
 public HashMap<RockLet, Boolean> getPool() {  
 return pool;  
 }  
  
}

