Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №1

**«Анализ кода»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

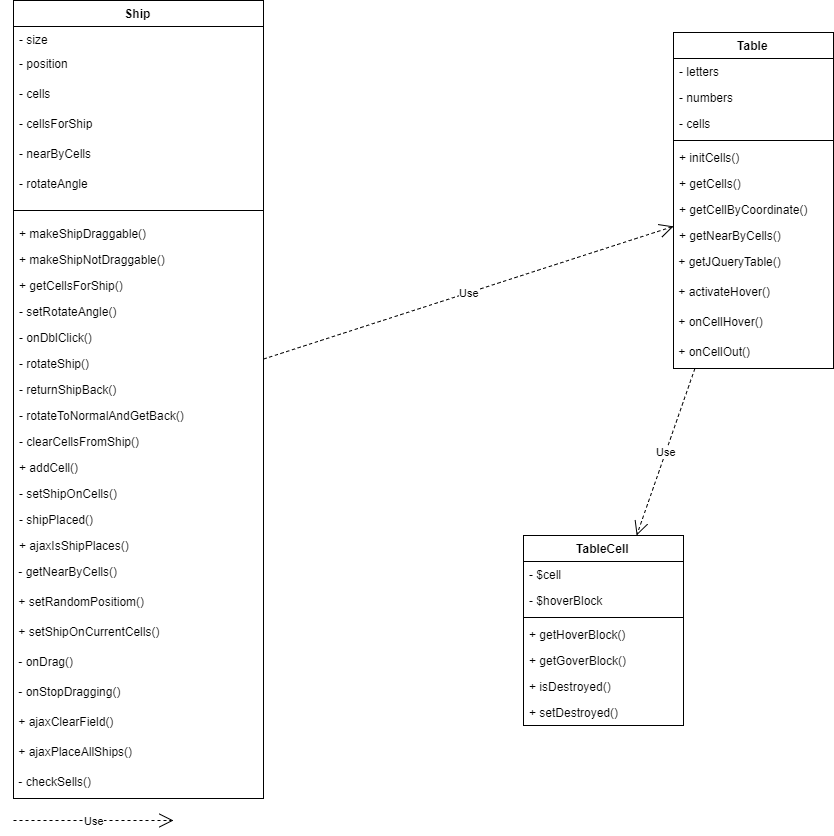
должность, ученое звание (и. о., фамилия)

Барнаул 2017

Задание:

* Опубликовать свой код проекта на github;
* Сделать uml-диаграмму к нему и выложить на github;
* Сделать code review на другие загруженные проекты.

**Диаграмма классов проекта, загруженного на github.**

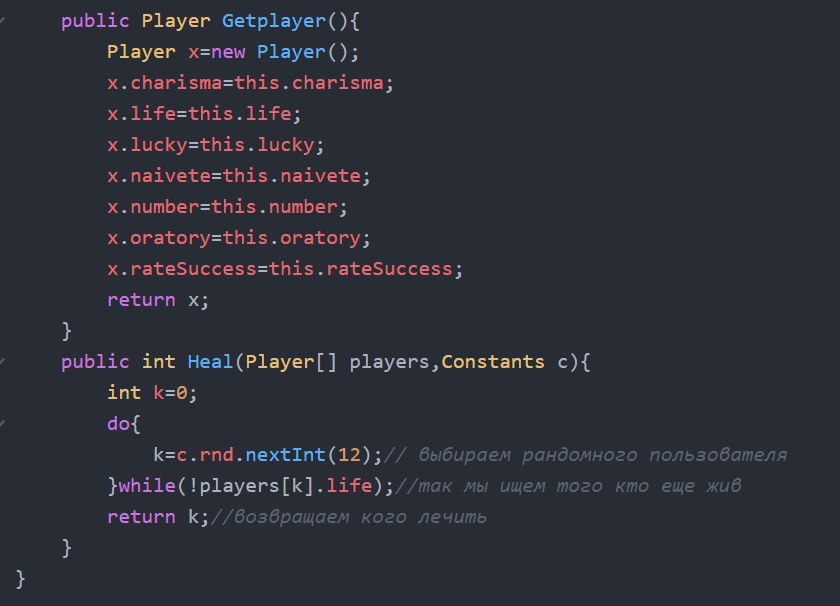


**Анализ кода.**

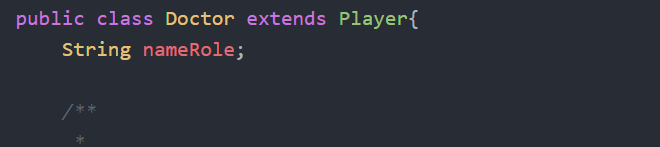
**Анализ №1.**

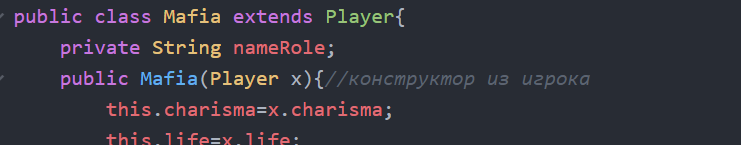
*Плюсы:*

1. Проект организован в виде классов
2. «Говорящие» названия классов, методов



1. Применяется наследование



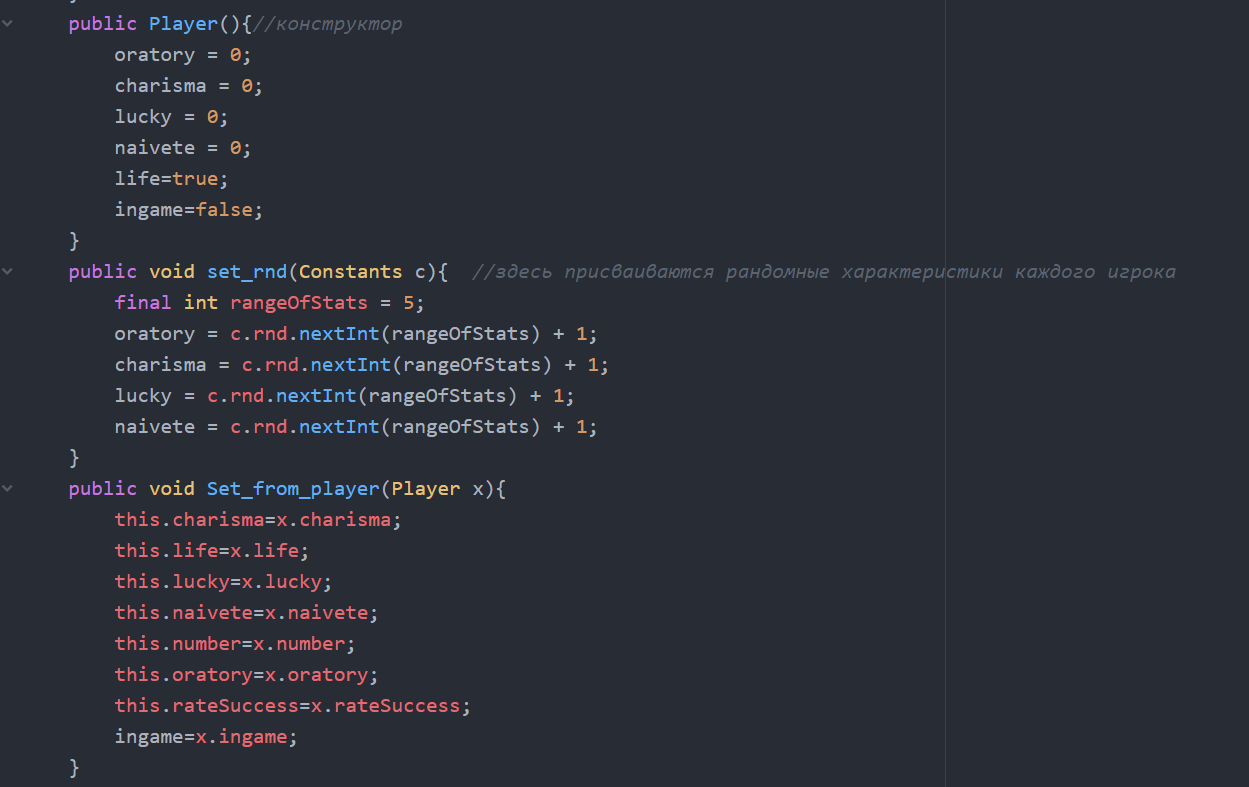


*Минусы:*

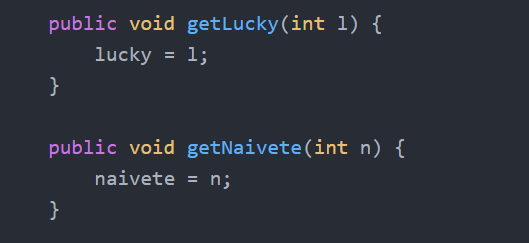
1. Мало комментариев. А те, что присутствуют, не информативны.
2. Не соблюдаются нормы форматирования кода.



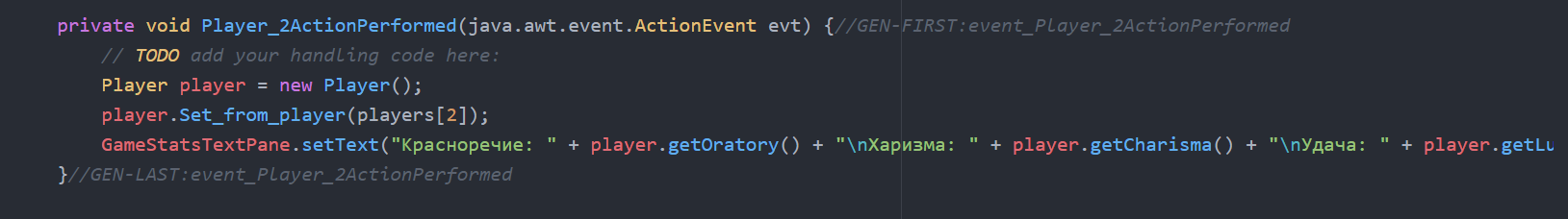
1. Нет единого стиля написания кода



1. Использование публичных переменных вместо геттеров. 
2. Написано get, но при этом метод ведет себя как сеттер.



1. Слишком длинные строки. Без горизонтальной прокрутки не видно весь код



Итоговая оценка:

3.7 из 10

**Анализ №2.**

Анализ кода программы FaceTracker

В ходе анализа исходного кода данной приложения были выделены следующие достоинства и недостатки.

Достоинства:

1. Организация проекта в виде классов.

2. В основном код выдержан в едином стиле написания.

3. Использованы стандартные структуры данных.

4. Использоваться разработанные и принятые сообществом сторонние компоненты.

5. Используются говорящие названия переменных и методов.

Недостатки:

1. Закомментированный участок кода.

2. Малоинформативное название массива объектов, вдобавок еще и не используется в дальнейшем.

3. Избыточная, в дальнейшем неиспользуемая переменная класса и соотв. ей метод.

4. Нерациональная загрузка файлов из директории, вместо использования файла ресурсов программы.

5. Несоблюдение общего стиля форматирования исходных данных.

6. Несоблюдение соглашения о именовании переменных.

7. Ошибка логического названия переменной. Называется age, хотя используется для хранения пути.

8. При инициализации объектов, в конструктор передается один и тот же ид ресурса со странным наименованием.

9. Низкая документированность проекта.

Подсчитаем оценку по формуле: оценка = кол−во плюсов / (кол−во плюсов+кол−во минусов) ∗10=514∗10≈3.6

Путем округления, получим итоговую оценку: 4

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №2

**«Паттерн Делегат»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

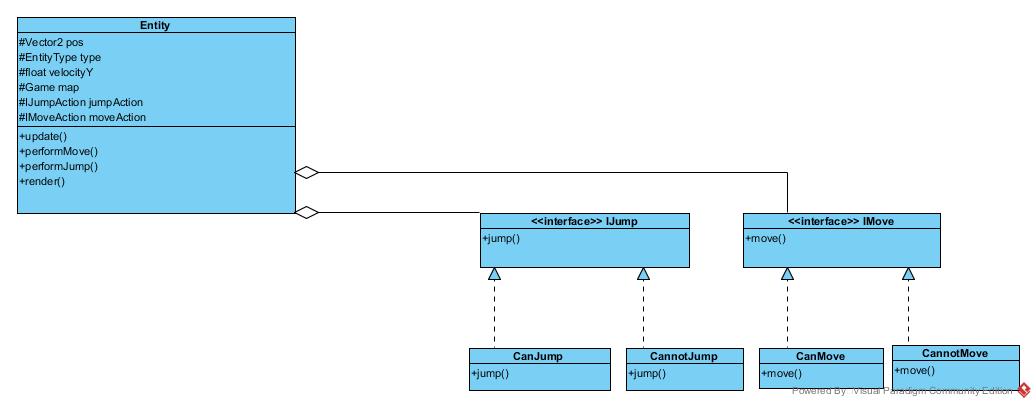
Барнаул 2017

Задание:

1. Рассмотреть задачу в неформальной постановке
2. Перечислить список возможных расширений постановки за дачи (новые свойства, действия, объекты, взаимодействия и т.д.)
3. Сформировать перечень интерфейсов
4. Сформировать перечень классов и их обязанности
5. Делегировать обязанности классам-делегатам
6. Построить диаграмму классов
7. Проверить выполнение принципов низкой связности и высокого зацепления.
8. Реализовать систему
9. Представить варианты реализации расширения системы

**Диаграмма.**

*Делегат.*



*Код реализации.*

public interface IJumpAction {

public void jump(Entity e, float amount);

}

public interface IMoveAction {

public void move(Entity e, GameMap map, float amount);

}

public class CannotMove implements IMoveAction {

@Override

public void move(Entity e, GameMap map, float amount) {

}}

public class CannotJump implements IJumpAction {

@Override

public void jump(Entity e, float amount) {

}

}

public class CanMove implements IMoveAction {

public void move(Entity e, GameMap map, float amount)

{

float newX = e.getPos().x + amount;

if (!map.doesRectCollideWithMap(newX, e.getPos().y, e.getWidth(), e.getHeight()))

e.setPosX(newX);

}

}

public class CanJump implements IJumpAction {

public void jump(Entity e, float amount){

e.setVelocityY(amount);

}

}

public abstract class Entity{

protected IJumpAction jumpAction;

protected IMoveAction moveAction;

….

public void performMove(float amount){

moveAction.move(this, map, amount);

}

public void performJump(float amount){

jumpAction.jump(this, amount);

}

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №3

**«Адаптер, Декоратор, Итератор, Композит»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

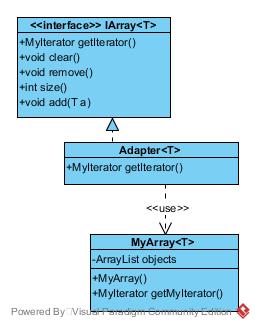
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

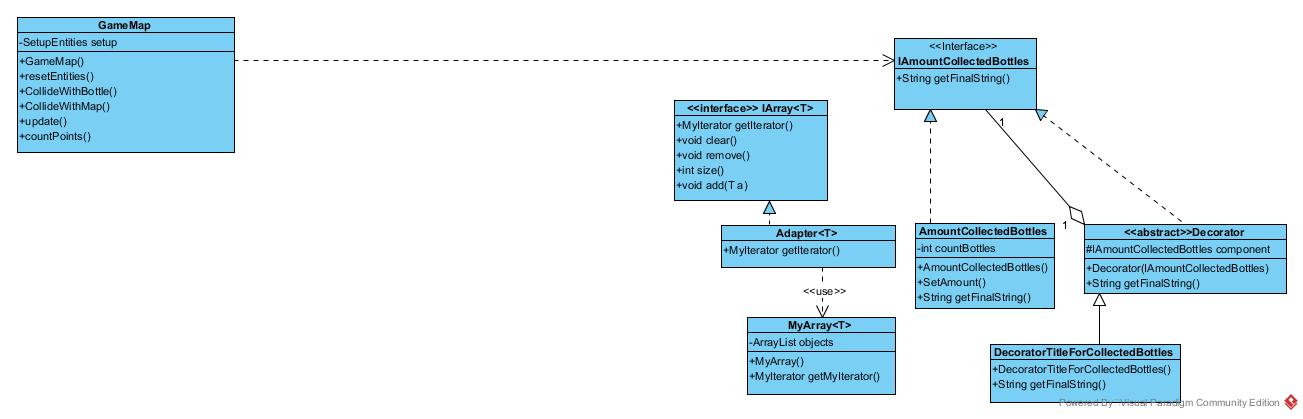
Барнаул 2017

**Диаграммы.**

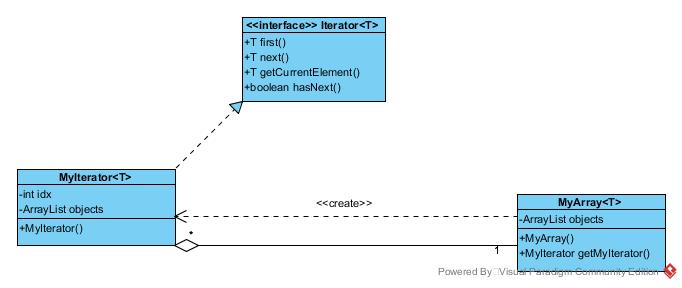
*Паттерн Адаптер.*

****

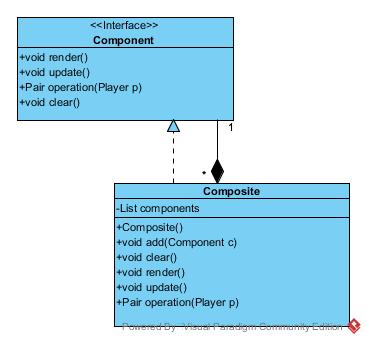
*Паттерн Декоратор.*

**

*Паттерн Итератор.*

**

*Паттерн Композит.*

**

**Код реализации.**

*Composite*

public interface Component {

public void render(SpriteBatch batch);

public void update(float delta, float gravity);

public Pair<Bottle,Boolean> operation(Player p);

public void clear();

}

public class Composite implements Component {

public List<Component> components;

public Composite(){

components = new ArrayList<Component>();

}

public Composite(List<Component> c){

components = c;

}

public void add(Component c){

components.add(c);

}

public void remove(Component c){

components.remove(c);

}

@Override

public void clear(){

for(Component c: components){

c.clear();

}

}

@Override

public void render(SpriteBatch batch) {

for(Component c: components){

c.render(batch);

}

}

@Override

public void update(float delta, float gravity) {

for(Component c: components){

c.update(delta, gravity);

}

}

@Override

public Pair<Bottle,Boolean> operation(Player p) {

Pair<Bottle,Boolean> f = new Pair<Bottle, Boolean>(null,false);

for(Component c: components){

f = c.operation(p);

if (f.getRight() == true)

break;

}

return f;}}

*Adapter*

public class Adapter<T> extends MyArray<T> implements IArray<T> {

@Override

public MyIterator<T> getIterator() {

return getMyIterator();

}

}

*Iterator*

public interface IArray<T> {

MyIterator<T> getIterator();

void clear();

void remove(int i);

void remove (T o);

void add (T o);

int size();

}

public interface Iterator<T> {

T first();

T next();

T getCurrrentElement();

boolean hasNext();

}

public class MyArray<T> {

ArrayList<T> objects;

public MyArray() {

objects = new ArrayList<T>();

}

public void add(T o){

objects.add(o);

}

public void remove(T o){

objects.remove(o);

}

public void clear(){objects.clear();}

public int size(){

return objects.size();

}

public void remove(int idx){

objects.remove(idx);

}

public MyIterator<T> getMyIterator() {

return new MyIterator<T>(objects);

}

}

public class MyIterator<T> implements Iterator<T> {

int idx = 0;

ArrayList<T> objects;

public MyIterator(ArrayList<T> objects) {

this.objects = objects;

}

@Override

public T first() {

idx = 0;

return objects.get(idx);

}

@Override

public T next() {

return objects.get(idx++);

}

@Override

public T getCurrrentElement() {

return objects.get(idx);

}

@Override

public boolean hasNext() {

return idx < objects.size();

}

}

*Декоратор*

public class AmountCollectedBottles implements IAmountCollectedBottles{

int countBottles;

public AmountCollectedBottles() {

countBottles = 0;

}

public void SetAmount(int c){

countBottles = c;

}

@Override

public String getFinalString() {

return String.valueOf(countBottles);

}

}

public abstract class Decorator implements IAmountCollectedBottles{

protected IAmountCollectedBottles component;

public Decorator(IAmountCollectedBottles c) {

this.component = c;

}

@Override

public String getFinalString() {

//TODO: Do decore bottle score

return component.getFinalString();

//

}

}

public class DecoratorTitleForCollectedBottles extends Decorator {

public DecoratorTitleForCollectedBottles(IAmountCollectedBottles c) {

super(c);

}

@Override

public String getFinalString() {

return "Bottles Scored: "+ super.getFinalString();

}

}

public interface IAmountCollectedBottles {

String getFinalString();

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №4

**«Мост, Фасад, Информационный эксперт, Приспособленец»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

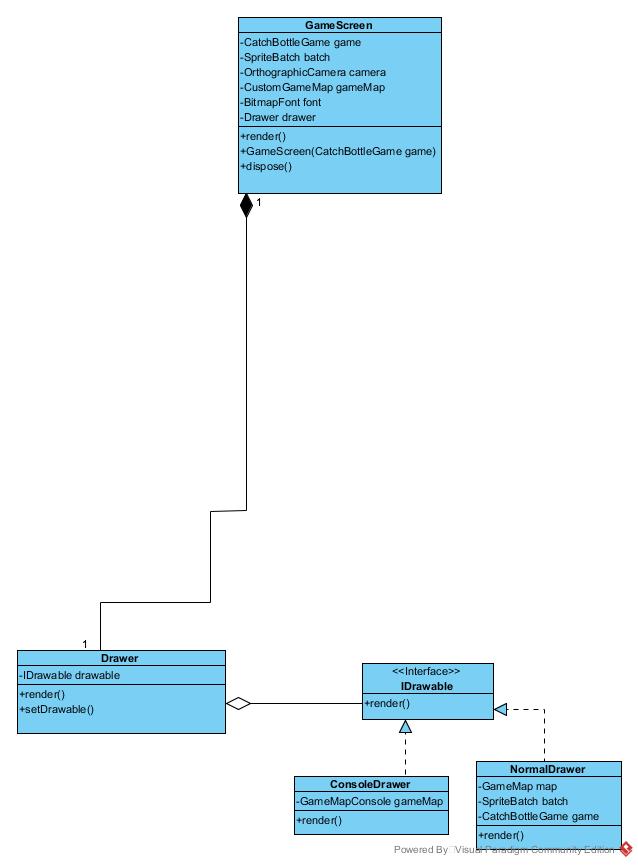
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

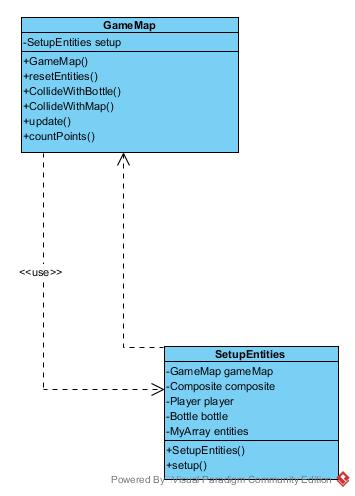
Барнаул 2017

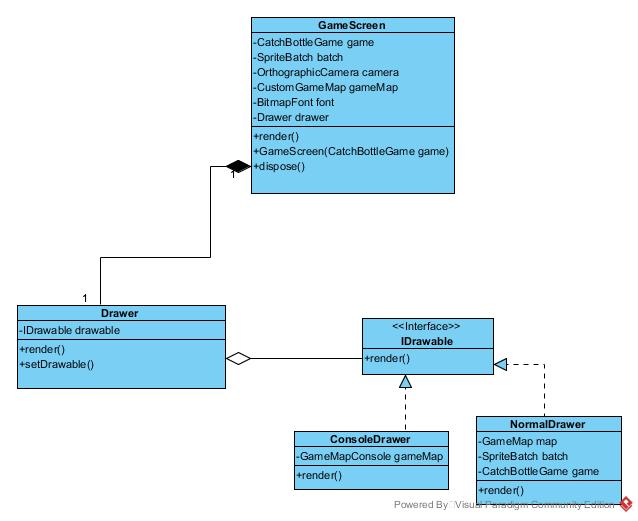
**Диаграммы.**

*Паттерн Мост.*

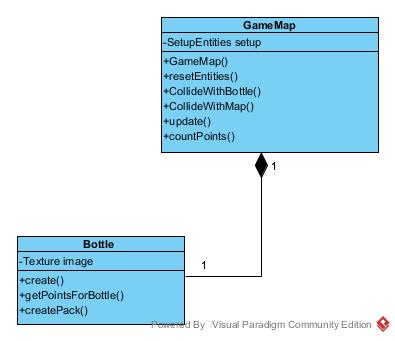


*Паттерн Фасад.*

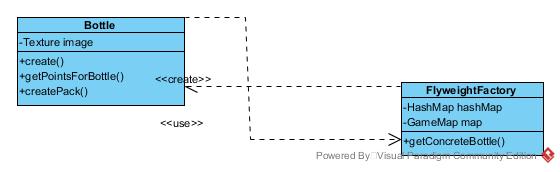
*.*

*Паттерн Мост*

*Паттерн Информационный Эксперт.*

**

*Паттерн Приспособленец.*



**Код реализации.**

*Паттерн Фасад.*

public class SetupEntities{

private GameMap map;

public IArray<Entity> entities = new Adapter<Entity>();

public Composite mainComposite = new Composite();

//TODO: FactoryMethod here

public Player player;

private IBottle bottle;

private IFactoryMethodBottle factoryMethod;

private Random rand = new Random(System.currentTimeMillis());

private Composite bottleComposite = new Composite();

private float bottleX;

public SetupEntities(GameMap map){

this.map = map;

}

public void setup(){

mainComposite.components.clear();

mainComposite.clear();

entities.clear();

player = new Player();

player.create(EntityType.PLAYER,new Vector2(470,464),map);

factoryMethod = new FactoryMethod();

bottle = factoryMethod.createBigBottle();

bottleX = rand.nextInt(1000-560)+560;

bottle.create(new Vector2(bottleX,464),map);

bottle.createPackBottles();

bottleComposite.add(bottle.getPack());

mainComposite.add((Component) bottle);

mainComposite.add(bottleComposite);

entities.add(player);

}

}

Паттерн Мост.

public class Drawer {

IDrawable drawable;

public Drawer(IDrawable drawable){

this.drawable = drawable;

}

public void render() throws InstantiationException, IllegalAccessException {

drawable.render();

}

public void setDrawable(IDrawable drawable){

this.drawable = drawable;

}

}

public class FakeConsoleDrawer implements IDrawable {

GameMapConsole gameMapConsole;

public FakeConsoleDrawer(GameMap map, OrthographicCamera camera) {

gameMapConsole = new GameMapConsole(map, camera);

}

@Override

public void render() {

gameMapConsole.update();

}

@Override

public void setState(IDrawable state) {

}

@Override

public IDrawable getSomeState(Class classState) {

return null;

}

}

public interface IDrawable {

void render() throws IllegalAccessException, InstantiationException;

void setState(IDrawable state);

IDrawable getSomeState(Class classState);

}

public class NormalDrawer implements IDrawable {

GameScreen.State state;

CatchBottleGame game;

SpriteBatch batch;

OrthographicCamera camera;

CustomGameMap gameMap;

BitmapFont font;

float fontW;

GlyphLayout layout;

float h;

MyButton menuButton;

IDrawable drawerState;

DrawerPause drawerPause;

DrawerDrawing drawerDrawing;

public NormalDrawer(GameScreen.State state, CatchBottleGame game, SpriteBatch batch, OrthographicCamera camera, CustomGameMap gameMap, BitmapFont font, float fontW, GlyphLayout layout, float h, MyButton menuButton) {

this.state = state;

this.game = game;

this.batch = batch;

this.camera = camera;

this.gameMap = gameMap;

this.font = font;

this.fontW = fontW;

this.layout = layout;

this.h = h;

this.menuButton = menuButton;

drawerPause = new DrawerPause(state,game,batch,camera,gameMap,font,fontW,layout,h,menuButton);

drawerDrawing = new DrawerDrawing(state,game,batch,camera,gameMap,font,fontW,layout,h,menuButton);

drawerState = drawerDrawing;

}

public IDrawable getSomeState(Class classState){

switch (classState.getSimpleName()){

case "DrawerPause":

return drawerPause;

case "DrawerDrawing":

return drawerDrawing;

default:

return null;

}

}

public void setState(IDrawable state){

drawerState = state;

}

@Override

public void render() throws IllegalAccessException, InstantiationException {

drawerState.render();

}

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №5

**«Фабрика, абстрактная фабрика, фабричный метод»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

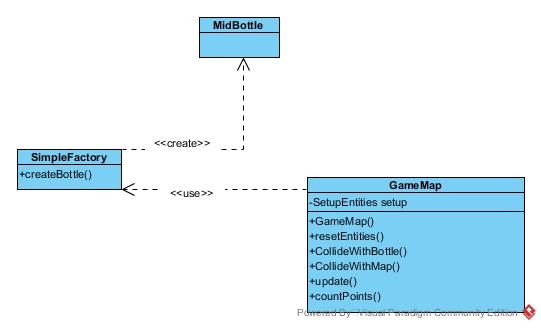
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

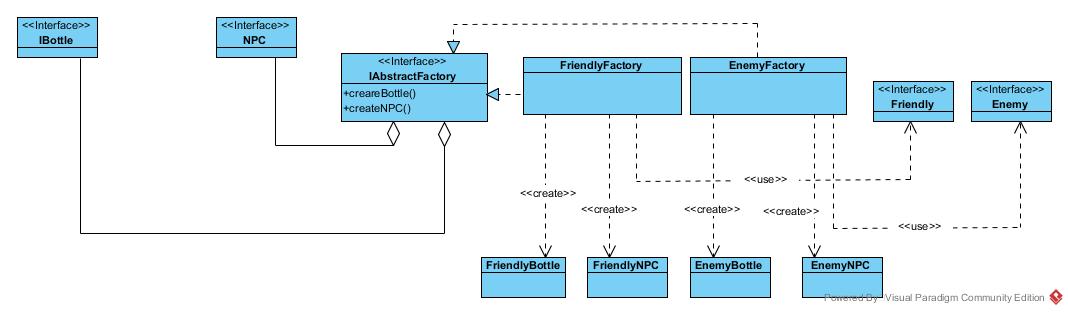
Барнаул 2017

**Диаграммы.**

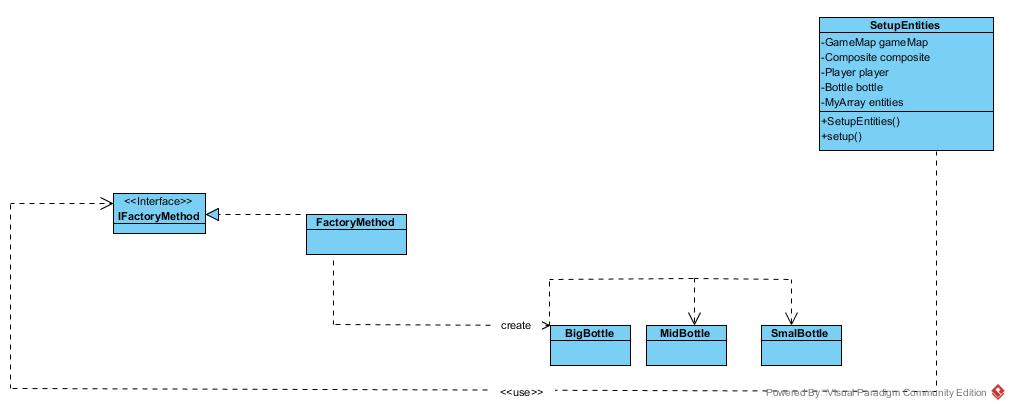
*Простая фабрика.*

**

*Абстрактная фабрика.*

**

*Фабричный метод.*

**

**Код реализации.**

*Простая фабрика.*

public class SimpleBottleFromSkyFactory {

PoolObjects<BottleFromSky> commonBottlePoolObjects;

PoolObjects<BottleFromSky> deadlyBottlePoolObjects;

public SimpleBottleFromSkyFactory(){

commonBottlePoolObjects = CommonBottlePool.getInstance();

deadlyBottlePoolObjects = DeadlyBottlePool.getInstance();

}

public BottleFromSky createBottle(Class o, GameMap m) {

switch (o.getSimpleName()) {

case "CommonBottle":

return commonBottlePoolObjects.checkOut(m);

case "DeadlyBottle":

return deadlyBottlePoolObjects.checkOut(m);

}

return null;

}

*Абстрактная фабрика.*

public interface IAbstractFactory {

NPC createNPC(GameMap map);

IBottle createBottle(GameMap map);

}

public class FriendlyFactory implements IAbstractFactory {

@Override

public NPC createNPC(GameMap map) {

return new FriendlyNPC(map);

}

@Override

public IBottle createBottle(GameMap map) {

return new FriendlyBottle(map);

}

}

public class EnemyFactory implements IAbstractFactory {

@Override

public NPC createNPC(GameMap map) {

return new EnemyNPC(map);

}

@Override

public IBottle createBottle(GameMap map) {

return new EnemyBottle(map);

}

}

*Фабричный метод.*

public interface IFactoryMethodBottle {

IBottle createBigBottle();

IBottle createMidBottle();

IBottle createSmallBottle();

}

public class FactoryMethod implements IFactoryMethodBottle {

@Override

public IBottle createBigBottle() {

return new BigBottle();

}

@Override

public IBottle createMidBottle() {

return new MidBottle();

}

@Override

public IBottle createSmallBottle() {

return new SmallBottle();

}

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №6

**«Строитель, синглтон, пул»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

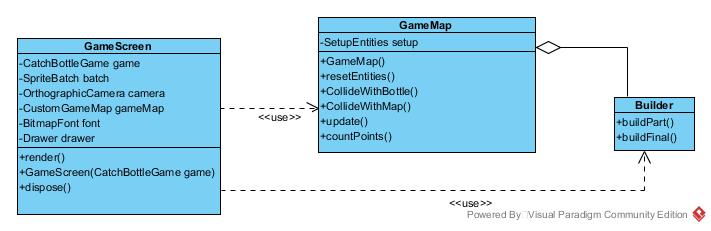
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

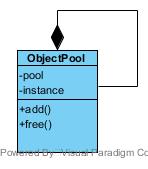
Барнаул 2017

**Диаграммы.**

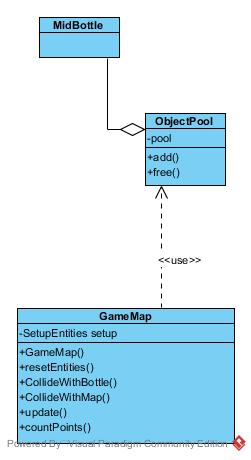
*Строитель.*



*Синглтон.*

**

*Пул.*

**

**Код реализации.**

*Строитель*

public static class Builder {

private int fontSize = 30;

private Color fontColor = Color.WHITE;

private Color fontBorderColor = Color.BLACK;

private int borderWidth = 2;

private int timeToGame = 0;

public Builder() {

}

public Builder setFontSize(int val) {

fontSize = val;

return this;

}

public Builder setTimeToGame(int val) {

timeToGame = val;

return this;

}

public Builder setFontColor(Color val) {

fontColor = val;

return this;

}

public Builder setFontBorderColor(Color val) {

fontBorderColor = val;

return this;

}

public Builder setBorderWidth(int val) {

borderWidth = val;

return this;

}

public CustomGameMap finalBuild() {

return new CustomGameMap(this);

}

}

*Синглтон*

public class CommonBottlePool extends PoolObjects<BottleFromSky> {

private static volatile CommonBottlePool singletonBottle = null;

public static CommonBottlePool getInstance() {

if (singletonBottle == null) {

singletonBottle = new CommonBottlePool();

}

return singletonBottle;

}

*Пул*

public abstract class PoolObjects<T extends BottleFromSky>{

public Set<T> available;

public Set<T> inUse;

protected abstract T create(GameMap map);

public T checkOut(GameMap map) {

if (available.isEmpty()) {

available.add(create(map));

}

Iterator<T> iterator = available.iterator();

T instance = iterator.next();

available.remove(instance);

inUse.add(instance);

// System.out.println("--------");

// System.out.println(available.size());

// System.out.println(inUse.size());

return instance;

}

public synchronized void checkIn(T instance) {

instance.resetPos();

available.add(instance);

inUse.remove(instance);

}

}

public class DeadlyBottlePool extends PoolObjects<BottleFromSky> {

private static volatile DeadlyBottlePool singletonBottle = null;

public static synchronized DeadlyBottlePool getInstance() {

if (singletonBottle == null) {

singletonBottle = new DeadlyBottlePool();

}

return singletonBottle;

}

@Override

protected DeadlyBottle create(GameMap map) {

return new DeadlyBottle(map);

}

private DeadlyBottlePool(){

available = new HashSet<>();

inUse = new HashSet<>();

}

}

public class CommonBottlePool extends PoolObjects<BottleFromSky> {

private static volatile CommonBottlePool singletonBottle = null;

public static CommonBottlePool getInstance() {

if (singletonBottle == null) {

singletonBottle = new CommonBottlePool();

}

return singletonBottle;

}

@Override

protected CommonBottle create(GameMap map) {

return new CommonBottle(map);

}

private CommonBottlePool(){

available = new HashSet<>();

inUse = new HashSet<>();

}

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №7

**«Наблюдатель, состояние, хранитель»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

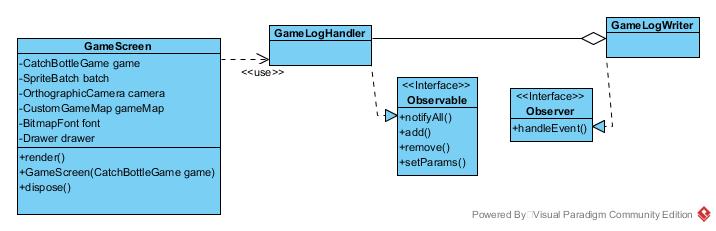
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

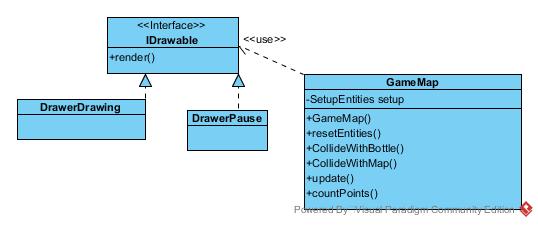
Барнаул 2017

**Диаграммы.**

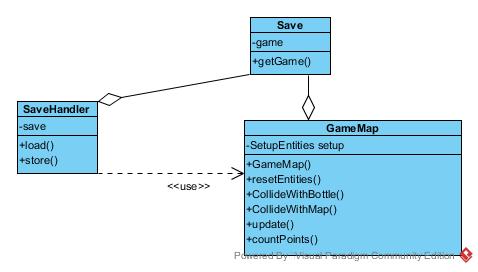
*Наблюдатель.*

**

*Состояние.*

**

*Хранитель.*

**

**Код реализации.**

*Наблюдатель.*

public interface Observer {

void handleEvent(long aliveTime, int score, int jumps);

}

public interface Observable {

void addObserver(Observer observer);

void removeObserver(Observer observer);

void notifyObservers();

void setParams(long time, int score, int jumps);

}

public class GameLogWriter implements Observer {

@Override

public void handleEvent(long aliveTime, int score, int jumps) {

System.out.println(String.format(">> [TIME ALIVE]: %d.0s\t[BOTTLES CATCHED]: %d\t[AMOUNT JUMPS]: %d <<", aliveTime,score,jumps));

}

}

public class GameLogHandler implements Observable {

long aliveTime;

int score;

int countJumps;

List<Observer> observers;

public GameLogHandler(){

observers = new ArrayList<>();

}

@Override

public void addObserver(Observer observer) {

observers.add(observer);

}

@Override

public void removeObserver(Observer observer) {

observers.remove(observer);

}

@Override

public void notifyObservers() {

for(Observer o: observers){

o.handleEvent(aliveTime,score,countJumps);

}

}

@Override

public void setParams(long time, int score, int jumps) {

aliveTime = time;

this.countJumps = jumps;

this.score = score;

notifyObservers();

}

}

*Хранитель.*

public class Save {

private int score;

public Save(int score){

this.score = score;

}

public int getScore() {

return score;

}

}

public class SaveHandler {

public static void saveToFile(String name, SetupEntities setupEntities, List<BottleFromSky> bottleFromSkies, int score) throws IOException {

Gdx.files.local(EntityLoader.PathToSaves + name + EntityLoader.ExtensionSaves).writeString("",false);

ArrayList<Entity> entities = new ArrayList<>();

entities.add(setupEntities.player);

entities.addAll(bottleFromSkies);

EntityLoader.saveEntities(name, entities,score);

}

public static void loadFromFile(String name, SetupEntities setupEntities, List<BottleFromSky> bottleFromSkies, GameMap map, int score) throws IOException {

ArrayList<Entity> entities = new ArrayList<>();

entities.addAll(EntityLoader.loadEntities(name, map, entities,score));

bottleFromSkies.clear();

for(Entity e: entities){

if ( e.getClass().getSuperclass().getGenericSuperclass() == Bottle.class){

BottleFromSky bs = null;

if (e.getType() == EntityType.COMMONBOTTLE) {

bs = new CommonBottle(map);

} else if (e.getType() == EntityType.DEADLYBOTTLE) {

bs = new DeadlyBottle(map);

}

if (bs == null)

continue;

bs.setPos(e.getPos());

bottleFromSkies.add(bs);

}

else

if(e.getClass() == Player.class){

setupEntities.player.setPos(e.getPos());

}

}

}

}

*Состояние.*

public class DrawerDrawing implements IDrawable {

GameScreen.State state;

CatchBottleGame game;

SpriteBatch batch;

OrthographicCamera camera;

CustomGameMap gameMap;

BitmapFont font;

float fontW;

GlyphLayout layout;

float h;

MyButton menuButton;

public DrawerDrawing(GameScreen.State state, CatchBottleGame game, SpriteBatch batch, OrthographicCamera camera, CustomGameMap gameMap, BitmapFont font, float fontW, GlyphLayout layout, float h, MyButton menuButton) {

this.state = state;

this.game = game;

this.batch = batch;

this.camera = camera;

this.gameMap = gameMap;

this.font = font;

this.fontW = fontW;

this.layout = layout;

this.h = h;

this.menuButton = menuButton;

}

@Override

public void render() throws IllegalAccessException, InstantiationException {

Gdx.gl.glClearColor(1, 0, 0, 1);

Gdx.gl.glBlendFunc(GL20.GL\_SRC\_ALPHA, GL20.GL\_ONE\_MINUS\_SRC\_ALPHA);

Gdx.gl.glClear(GL20.GL\_COLOR\_BUFFER\_BIT);

gameMap.render(camera, batch);

game.batch.begin();

menuButton.draw(game.batch);

game.batch.end();

}

@Override

public void setState(IDrawable state) {

}

@Override

public IDrawable getSomeState(Class classState) {

return null;

}

}

public class DrawerPause implements IDrawable{

GameScreen.State state;

CatchBottleGame game;

SpriteBatch batch;

OrthographicCamera camera;

CustomGameMap gameMap;

BitmapFont font;

float fontW;

GlyphLayout layout;

float h;

MyButton menuButton;

public DrawerPause(GameScreen.State state, CatchBottleGame game, SpriteBatch batch, OrthographicCamera camera, CustomGameMap gameMap, BitmapFont font, float fontW, GlyphLayout layout, float h, MyButton menuButton) {

this.state = state;

this.game = game;

this.batch = batch;

this.camera = camera;

this.gameMap = gameMap;

this.font = font;

this.fontW = fontW;

this.layout = layout;

this.h = h;

this.menuButton = menuButton;

}

@Override

public void render() throws IllegalAccessException, InstantiationException {

game.batch.begin();

font.draw(game.batch,layout,(camera.viewportWidth - fontW )/2,h/2);

game.batch.end();

}

@Override

public void setState(IDrawable state) {

}

@Override

public IDrawable getSomeState(Class classState) {

return null;

}

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №8

**«Команда, перенаправление, посетитель»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

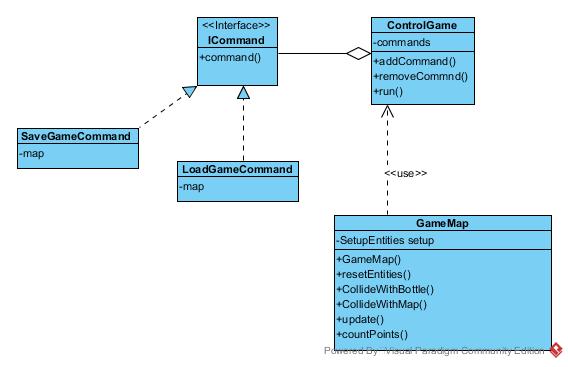
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

Барнаул 2017

**Диаграммы.**

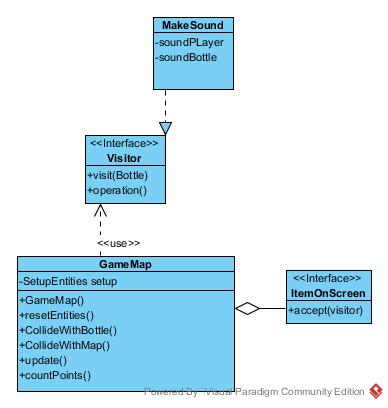
*Команда.*

**

*Перенаправление.*

**

*Посетитель.*

**

Код реализации.

*Команда.*

public interface ICommand {

void command(String filename) throws IOException;

}

public class ControlGame {

List<ICommand> commands;

String filename;

public ControlGame(String filename) {

commands = new ArrayList<>();

this.filename = filename;

}

public void addCommand(ICommand c) {

commands.add(c);

}

public void removeCommand(ICommand c) {

commands.remove(c);

}

public void run() throws IOException {

for (ICommand c : commands) {

c.command(filename);

}

commands.clear();

}

}

public class OpenGameCommand implements ICommand {

private GameMap map;

public OpenGameCommand(GameMap map){

this.map = map;

}

@Override

public void command(String filename) throws IOException {

map.load(filename);

}

}

public class SaveGameCommand implements ICommand{

private GameMap map;

public SaveGameCommand(GameMap map){

this.map = map;

}

@Override

public void command(String filename) throws IOException {

map.save(filename);

}

}

*Перенаправление.*

public class CustomGameMapLoader {

private static Json json = new Json();

private static final int SIZE = 100;

public static CustomGameMapData loadMap (String id) {

FileHandle file = Gdx.files.local("maps/" + id + ".map");

CustomGameMapData data = json.fromJson(CustomGameMapData.class, file.readString());

return data;

}

public static void saveMap (String id, int[][][] map) {

CustomGameMapData data = new CustomGameMapData();

data.id = id;

data.name = id;

data.map = map;

Gdx.files.internal("maps/").file().mkdirs();

FileHandle file = Gdx.files.local("maps/" + id + ".map");

file.writeString(json.prettyPrint(data), false);

}

public static void saveMap (CustomGameMapData data) {

Gdx.files.internal("maps/").file().mkdirs();

FileHandle file = Gdx.files.local("maps/" + data.id+ ".map");

file.writeString(json.prettyPrint(data), false);

}

public static CustomGameMapData createTiledMap (String id) {

CustomGameMapData mapData = new CustomGameMapData();

mapData.id = id;

mapData.name = id;

mapData.map = new int[2][SIZE][SIZE];

Random random = new Random();

for (int row = 0; row < SIZE; row++) {

for (int col = 0; col < SIZE; col++) {

mapData.map[0][row][col] = TileType.SKY.getId();

if (row > SIZE - 3 - 1) {

mapData.map[1][row][col] = TileType.SKY.getId();

} else if (row > SIZE - 20) {

mapData.map[1][row][col] = TileType.STONE.getId();

} else if (row > SIZE - 25) {

mapData.map[1][row][col] = TileType.STONE.getId();

} else if (row > SIZE - 30) {

mapData.map[1][row][col] = TileType.STONE.getId();

} else {

if (random.nextInt(50) == 0)//1 chance out of 50 of being a cloud

mapData.map[1][row][col] = TileType.BOTTLE.getId();

}

}

}

return mapData;

}

}

*Посетитель.*

public interface ItemOnScreen {

void accept(Visitor visitor);

}

public interface Visitor {

void visit(Bottle bottle);

void visit(Player player);

}

public class MakeSoundByEntity implements Visitor {

Sound bottleSound;

Sound playerSound;

final static float volume = 0.2f;

public MakeSoundByEntity(){

bottleSound = Gdx.audio.newSound(Gdx.files.internal("Sounds/bottleSound.wav"));

playerSound = Gdx.audio.newSound(Gdx.files.internal("Sounds/playerSound.wav"));

}

@Override

public void visit(Bottle bottle) {

bottleSound.play(volume);

}

@Override

public void visit(Player player) {

playerSound.play();

}

}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

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

Преподаватель С. М. Старолетов\_

(подпись) (и. о.,фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

дата

Отчет

По лабораторной работе №9

**«Цепочка, медиатор, стратегия»**

по дисциплине «Архитектурное проектирование и паттерны программирования»

ЛР 09.03.04.01.000 O

Студент группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПИ-42\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. Е. Баев

(и. о., фамилия)

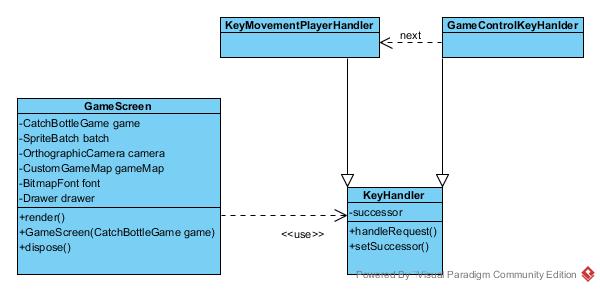
Преподаватель\_\_\_\_\_\_\_доцент, к. ф.-м. н. \_\_\_\_\_\_\_\_\_ С. М. Старолетов

должность, ученое звание (и. о., фамилия)

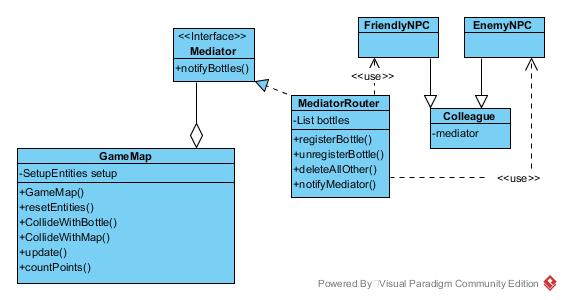
Барнаул 2017

**Диаграммы.**

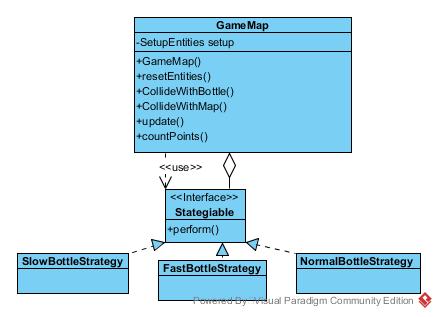
*Цепочка.*



*Медиатор.*

**

*Стратегия.*

**

**Код реализации.**

*Цепочка обязанностей.*

public abstract class KeyHandler {

public KeyHandler successor;

public abstract void handleRequest(int keycode, GameMap map) throws IOException;

public KeyHandler(KeyHandler successor){

this.successor = successor;

}

public KeyHandler(){

}

public KeyHandler getSuccessor() {

return successor;

}

public void setSuccessor(KeyHandler successor) {

this.successor = successor;

}

}

public class PlayerMovementHandler extends KeyHandler {

public PlayerMovementHandler(){

super();

}

public PlayerMovementHandler(KeyHandler successor) {

super(successor);

}

@Override

public void handleRequest(int keycode, GameMap map) throws IOException {

switch (keycode){

case Input.Keys.SPACE:

case Input.Keys.UP:

map.sEntities.player.playerStateMoving = Player.PlayerStateMoving.JUMP;

break;

case Input.Keys.LEFT:

map.sEntities.player.playerStateMoving = Player.PlayerStateMoving.MOVE\_L;

break;

case Input.Keys.RIGHT:

map.sEntities.player.playerStateMoving = Player.PlayerStateMoving.MOVE\_R;

break;

default:

if(successor!=null)

successor.handleRequest(keycode, map);

}

}

}

public class GameControlHandler extends KeyHandler {

final String filename = "save\_1";

public GameControlHandler() {

super();

}

public GameControlHandler(KeyHandler successor) {

super(successor);

}

@Override

public void handleRequest(int keycode, GameMap map) throws IOException {

switch (keycode) {

case Input.Keys.S:

map.save(filename);

break;

case Input.Keys.L:

map.load(filename);

break;

default:

if (successor != null)

successor.handleRequest(keycode, map);

}

}

}

*Стратегия.*

public abstract class Strategy implements Strategiable {

float gravityScale;

List<BottleFromSky> bottles;

public Strategy(List<BottleFromSky> bottleList){

this.bottles = bottleList;

}

@Override

public void perform() {

for (Bottle b : bottles) {

b.setVelocityY(b.getVelocityY() \* gravityScale);

}

}

}

public class SlowBottleStrategy extends Strategy {

public SlowBottleStrategy(List<BottleFromSky> bottleList) {

super(bottleList);

gravityScale = 0.1f;

}

}

public class NormalBottleStrategy extends Strategy {

public NormalBottleStrategy(List<BottleFromSky> bottleList) {

super(bottleList);

gravityScale = 1.0f;

}

@Override

public void perform() {

for (Bottle b : bottles) {

b.setVelocityY(-3.0f);

}

}

}

public class FastBottleStrategy extends Strategy {

public FastBottleStrategy(List<BottleFromSky> bottleList) {

super(bottleList);

gravityScale = 2.0f;

}

}

*Медиатор.*

public class MediatorRouter implements Mediator{

HashSet<Colleague> colleagues = new HashSet<>();

public void add(Colleague colleague) {

colleagues.add(colleague);

}

public void remove(Colleague colleague) {

colleagues.remove(colleague);

}

public HashSet<Colleague> get(){

return colleagues;

}

@Override

public void send(Colleague e) {

for(Colleague colleague: colleagues) {

if(colleague != e) {

colleague.receive(e);

}

}

}

}

public interface Mediator {

void send(Colleague e);

}

public interface Colleague {

void receive(Colleague c);

void send();

}