Факультет мехатроники и робототехники

Лабораторная работа №3

По дисциплине : «Программирование»

Вариант: 317306

Преподаватель: Райла Мартин

Выполнил: Нуорузи Мехди

Группа: R3137

Санкт-Петербург

2020

# Задание:

**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().
4. Программа должна содержать как минимум один перечисляемый тип (enum).

**Порядок выполнения работы:**

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов объектной модели.
3. Исходный код программы.
4. Результат работы программы.
5. Выводы по работе.

# Описание предметной области:

Введите вариант: 317306

Описание предметной области, по которой должна быть построена объектная модель:

Коротышки, держась за протянутую поперек двора веревку, принялись таскать Шпунтику все, что могло понадобиться для приготовления обеда. Все работали очень активно, так как каждому очень хотелось есть. Не работали только больной Винтик да еще Пончик, который по-прежнему болтался под потолком в столовой. Знайка сказал, что Пончик, очевидно, потерял ориентацию в пространстве и не сумел приспособиться к состоянию невесомости. На самом же деле Пончик прекрасно приспособился к невесомости, но так как он был чрезвычайно хитрый, то решил это скрыть. В то время, как все коротышки работали, он летал потихоньку по комнате и уплетал манную кашу, которая вывалилась из кастрюли и плавала вокруг комьями. За небольшой промежуток времени он единолично съел целую кастрюлю каши, так что от нее и следа не осталось. Пока коротышки варили себе обед, Знайка привязался к веревке и производил во дворе наблюдения над силой тяжести. Оказалось, что состояние невесомости наблюдалось вокруг дома только на расстоянии двадцати тридцати шагов. Это была, как ее назвал Знайка, зона невесомости. За ней начиналась, как ее назвал Знайка, зона тяжести, или зона весомости. Пробравшись через зону невесомости при помощи веревки, можно было проникнуть в зону весомости и, выйдя из калитки, уже без всяких опасений отправляться в любом направлении по улице.

The short men, holding on to a rope stretched across the yard, began to drag Shpuntik with everything that might be needed to prepare dinner.

Everyone worked very actively, as everyone really wanted to eat.

The only ones not working were the ailing Screw and the Donut, which was still hanging from the ceiling in the dining room.

Znayka said that Ponchik apparently lost his orientation in space and failed to adapt to the state of weightlessness.

In fact, the Donut perfectly adapted to weightlessness, but since he was extremely cunning, he decided to hide it.

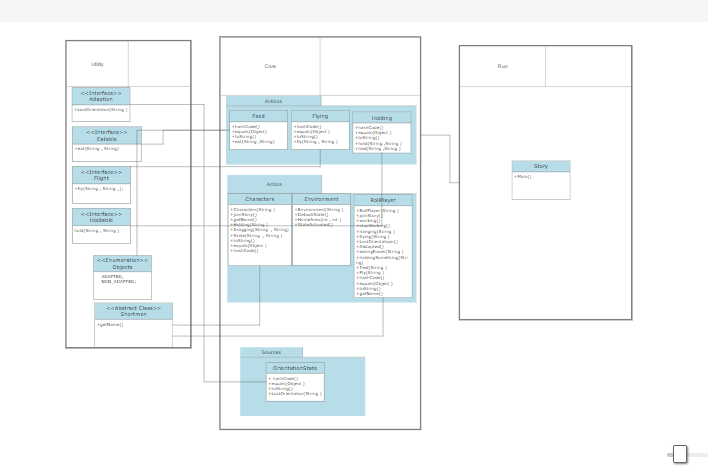
While all the shorties were working, he flew slowly around the room and ate semolina, which fell out of the pan and floated around in clods.

In a short period of time, he single-handedly ate a whole pot of porridge, so that there was no trace of it. While the short men were cooking dinner for themselves, Znayka tied himself to a rope and made observations on gravity in the yard.

It turned out that the state of weightlessness was observed around the house only at a distance of twenty-thirty steps. It was, as Znayka called it, a weightless zone.

Behind it began, as Znayka called it, a zone of gravity, or a zone of weight.

Having made his way through the zone of weightlessness with the help of a rope, it was possible to penetrate into the zone of gravity and, leaving the gate, without any fear to go in any direction along the street.



# Исходный код:

Story.java - класс отвечающий за работу программы.

package run;

import core.actors.Characters;

import core.actors.Environment;

import core.actors.RollPlayer;

public class Story {

public static void main(String[] args) {

try {

Characters character = new Characters("Shorties");

Environment environment = new Environment("weightlessness");

character.Holding("Rope");

environment.HomeArea(20,30);

environment.StateActivated();

character.Dragging("Shpuntik ", "for preparing dinner. ");

character.State(" Working ", " for eating (Hunger) ");

RollPlayer screw = new RollPlayer("Screw");

RollPlayer donut = new RollPlayer("donut");

screw.stopWorking();

donut.stopWorking();

screw.hanging("roof");

donut.hanging("roof");

RollPlayer znayka = new RollPlayer("znayka");

RollPlayer ponchik = new RollPlayer("ponchik");

znayka.LostOrientatopn();

ponchik.LostOrientatopn();

donut.Addapted();

donut.flying("Room");

donut.eatingFoods("semolina");

donut.eatingFoods("porridge");

znayka.holdingSomething("rope");

znayka.Tied("rope");

znayka.Fly("gravity zone");

} catch (Exception e) {

System.out.println("It doesn't variable");

}

}

}

2.2. core package

Characters.java – класс «Characters» - объект, совершающий действия в программе.

package core.actors;

package core.actors;

import utility.Shortmen;

import java.util.Objects;

public class Characters extends Shortmen {

private String name;

public Characters(String name) {

this.name = name;

joinStory();

}

private void joinStory() {

System.out.println(name + " join in story");

}

@Override

public String getName() {

return name;

}

public void Holding(String \_rope){

System.out.println(name + " holding " + \_rope);

}

public void Dragging(String \_object , String \_Resone){

System.out.println(name + " dragging " + \_object + "" + \_Resone);

}

public void State(String \_state , String \_Resone){

System.out.println(name + \_state + " " + \_Resone);

}

@Override

public String toString() {

return "Man{" +

"name='" + name + '\'' +

'}';

}

@Override

public boolean equals(Object object) {

if (this == object) return true;

if (object == null || getClass() != object.getClass()) return false;

Characters man = (Characters) object;

return Objects.equals(name, man.name);

}

@Override

public int hashCode() {

return Objects.hash(name);

}

}

RollPlayer.java – класс «RollPlayer» - объект, совершающий действия в программе.

package core.actors;

import core.actions.Holding;

import core.actions.Feed;

import core.actions.Flying;

import core.sources.OrientationState;

import utility.Flight;

import utility.Shortmen;

public class RollPlayer extends Shortmen {

private String name;

public RollPlayer(String name) {

this.name = name;

joinStory();

}

private void joinStory() {

System.out.println(name + " join in story");

}

public void working() {

System.out.println(name + " start working");

}

public void stopWorking() {

System.out.println(name + " stop working");

}

public void hanging(String \_from) {

System.out.println(name + " hangs from " + \_from);

}

public void flying(String \_where) {

System.out.println(name + " flying in " + \_where);

}

public void LostOrientatopn() {

OrientationState orientationState = new OrientationState();

orientationState.LostOrientaton(name);

}

public void Addapted() {

OrientationState orientationState = new OrientationState();

orientationState.AdaptedOrientaton(name);

}

public void eatingFoods(String \_food){

Feed action = new Feed();

action.eat(name, \_food);

}

public void holdingSomething(String \_obj){

Holding action = new Holding();

action.hold(name,\_obj);

}

public void Tied(String \_obj){

Holding action = new Holding();

action.tied(name,\_obj);

}

public void Fly(String \_destnation){

Flying action = new Flying();

action.fly(name,\_destnation);

}

@Override

public int hashCode() {

return super.hashCode();

}

@Override

public boolean equals(Object obj) {

return super.equals(obj);

}

public String toString() {

return name +" was a famous astronomer";

}

@Override

public String getName() {

return name;

}

}

2.3. utility package

**package** utility;

**public** **interface** Adaption {

**void** LostOrientaton(String name);

**void** AdaptedOrientaton(String name);

}

**package** utility;

**public** **enum** CharState {

***Tied***,

***NotTied***

}

**package** utility;

**public** **interface** Eatable {

**void** eat(String \_name, String \_food);

}

**package** utility;

**public** **interface** Flight {

**void** fly(String \_name, String \_destenation);

}

**package** utility;

**public** **interface** Holdable {

**void** hold(String name, String \_obj);

**void** tied(String \_name, String \_obj);

}

**package** utility;

**public** **abstract** **class** Shortmen {

**public** **abstract** String getName();

**public** **abstract** **void** setName(String name) **throws** WrongNameException;

}

**package** utility;

**public** **enum** Objects {

***ADAPTED***,

***NON\_ADAPTED***;

}

# Результат работы:

Shorties join in story

weightlessness Inactive

Shorties holding Rope

weightlessness mode at a distance of 20 to 30 steps has been set.

weightlessness Activated

Shorties dragging Shpuntik for preparing dinner.

Shorties Working for eating (Hunger)

Screw join in story

donut join in story

Screw stop working

donut stop working

Screw hangs from roof

donut hangs from roof

znayka join in story

ponchik join in story

znayka Lost his orientaton

ponchik Lost his orientaton

donut Adapted in space.

donut flying in Room

donut eats semolina

donut eats porridge

znayka holds rope

znayka tied himself to rope

znayka is flying to gravity zone

# Вывод:

В этой лабораторной работе я научился применять принципы SOLID на практике, применять соглашения об именах Java, более внимательно изучать интерфейсы, абстрактные классы и перечисления.

In this lab, I learned how to put the principles of SOLID in practice, apply Java naming conventions, take a closer look at interfaces, abstract classes, and enumerations