Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 3-4**

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант №55665-55665.524

*Выполнил:*

Студент группы P3112

Дениченко Александр Олегович

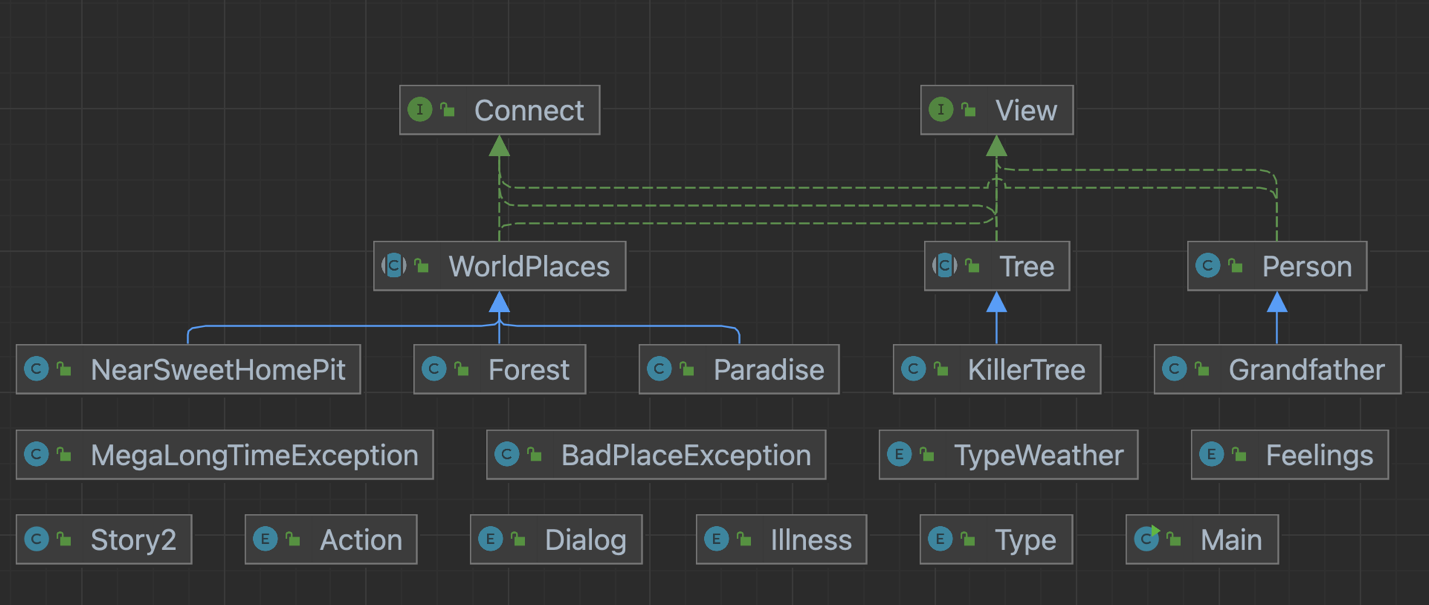
*Преподаватель:*

Гаврилов Антон Валерьевич



Санкт-Петербург, 2022

# Диаграмма классов



# Описание предметной области

Был чудесный зимний день. Пятачок, разметавший снег у дверей своего дома, поднял голову и увидел не кого иного, как Винни-Пуха. Пух медленно шел куда-то, внимательно глядя себе под ноги, и так глубоко задумался, что, когда Пятачок окликнул его, он и не подумал остановиться. Пятачок подошел к нему поближе: Он замолчал и решительно зашагал вперед по следу, а Пятачок, помедлив минутку-другую, побежал за ним. Внезапно Винни-Пух остановился и нагнулся к земле. Пятачок мужественно почесал за ухом и сказал, что до пятницы он совершенно свободен и с большим удовольствием пойдет с Пухом, в особенности если там Настоящий Бука. И они пошли дальше вместе. Следы шли вокруг маленькой ольховой рощицы... и, значит, два Буки, если это были они, тоже шли вокруг рощицы, и, понятно, Пух и Пятачок тоже пошли вокруг рощицы. По пути Пятачок рассказывал Винни-Пуху интересные истории из жизни своего дедушки Посторонним В. Например, как этот дедушка лечился от ревматизма после охоты и как он на склоне лет начал страдать одышкой, и всякие другие занятные вещи. А Пух все думал, как же этот дедушка выглядит. И ему пришло в голову, что вдруг они сейчас охотятся как раз на двух дедушек, и интересно, если они поймают этих дедушек, можно ли будет взять хоть одного домой и держать его у себя, и что, интересно, скажет по этому поводу Кристофер Робин. А следы все шли и шли перед ними... Вдруг Винни-Пух снова остановился как вкопанный. И они пошли дальше, начиная немного волноваться, потому что ведь эти три Неизвестных Зверя могли оказаться Очень Страшными Зверями. И Пятачку ужасно хотелось, чтобы его милый Дедушка Посторонним В. был бы сейчас тут, а не где-то в неизвестном месте... А Пух думал о том, как было бы хорошо, если бы они вдруг, совсем-совсем случайно, встретили Кристофера Робина,-- конечно, просто потому, что он, Пух, так любит Кристофера Робина!.. И тут совершенно неожиданно Пух остановился в третий раз и облизал кончик своего носа, потому что ему вдруг стало страшно жарко. Перед ними были следы четырех зверей!

# Исходный код

Package Exceptions

BadPlaceException:

package Exceptions;  
  
public class BadPlaceException extends NullPointerException{  
  
 public BadPlaceException() {  
 super("place is null");  
 }  
  
 @Override  
 public String toString() {  
 return "\n Ошибка, место не может иметь значение null!\n";  
 }  
}

MegaLongTimeException:

package Exceptions;  
  
public class MegaLongTimeException extends Exception{  
 public MegaLongTimeException() {  
 super("Too long");  
 }  
  
 @Override  
 public String toString() {  
 return "\n Ошибка, животное не может гулять более 24 часов!\n";  
 }  
}

Package intefaces:

Connect:

package interfaces;  
  
public interface Connect {  
 void join();  
 void disconnect();  
 void request();  
}

View:

package interfaces;  
  
public interface View {  
 void appear();  
 void outlook();  
}

Package other

KillerTree:

package other;  
import interfaces.\*;  
import people.\*;  
  
public class KillerTree extends Tree {  
 public static class Apple implements Connect, View{  
 public void fallToPerson(Person goal){  
 System.*out*.println("Дерево трещит и яблоко начинает быстро падать на "+goal.getName()+" и убивает его на смерть!");  
 goal.setFeelings(Feelings.*NORMAL*);  
 goal.setAct(Action.*NOTHING*);  
 }  
 @Override  
 public void join() {  
 System.*out*.println("Яблоко выросло на дереве");  
 }  
  
 @Override  
 public void disconnect() {  
 System.*out*.println("Яблоко сгнило и пропало");  
 }  
  
 @Override  
 public void request() {  
 System.*out*.println("Вырастет ли яблоко, пока загадка");  
 }  
  
 @Override  
 public void appear() {  
 System.*out*.println("Яблоко переливается на солнышке");  
 }  
  
 @Override  
 public void outlook() {  
 System.*out*.println("Падающие яблоки делали все эти неизвестные следы, это самые жуткие убийцы в лесах. Их боялись даже буки");  
 }  
 }  
 @Override  
 public void join() {  
 System.*out*.println("Дерево резко выросло");  
 }  
  
 @Override  
 public void disconnect() {  
 System.*out*.println("Дерево упало и пропало");  
 }  
  
 @Override  
 public void request() {  
 System.*out*.println("Есть вероятность, что дерево выростет");  
 }  
  
 @Override  
 public void appear() {  
 System.*out*.println("Дерево устрашающе шелестит листочками");  
 }  
  
 @Override  
 public void outlook() {  
 System.*out*.println("Это дерево похоже на убийцу");  
 }  
}

Tree:

package other;  
  
import interfaces.\*;  
  
public abstract class Tree implements Connect, View {  
 @Override  
 public void join() {  
  
 }  
  
 @Override  
 public void disconnect() {  
  
 }  
  
 @Override  
 public void request() {  
  
 }  
  
 @Override  
 public void appear() {  
  
 }  
  
 @Override  
 public void outlook() {  
  
 }  
}

Package people:

Action:

package people;  
  
public enum Action {  
 *NOTHING*("ничего не делает",false),  
 *WALK*("гуляет",true),  
 *SEARCH\_FOR\_PREY*("ищут добычу",false),  
 *KILL*("убивают",true),  
 *CLEAN\_SNOW*("убирает снег",true);  
 private String name;  
  
 private boolean takeStrength;  
 Action(String name, boolean takeStrength){  
 this.name = name;  
 this.takeStrength = takeStrength;  
 }  
  
 public boolean getTakeStrength() {  
 return takeStrength;  
 }  
  
 public void setTakeStrength(boolean takeStrength) {  
 this.takeStrength = takeStrength;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

Dialog:

package people;  
  
public enum Dialog {  
 *SPEAKER*("говорит", "теряет силы говорения"),  
 *LISTENER*("слушает", "восполняет силы языка"),  
 *VOID*("молчит", "развивает социофобию");  
 private String name;  
 private String description;  
 Dialog(String name, String description){  
 this.name = name;  
 this.description= description;  
 }  
  
 public String getName() {  
 return name;  
 }  
 public String getDescription(){return description;}  
 public void setDescription(String name) {  
 this.description = description;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
}

Feelings:

package people;  
  
public enum Feelings {  
 *BAD*("злится"),  
 *JOY*("веселится"),  
 *AFRAID*("встревожен"),  
 *NORMAL*("не беспкоится");  
 private String name;  
 Feelings(String name){  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

Grandfather:

package people;  
import places.\*;  
  
public class Grandfather extends Person {  
  
 public Grandfather(String name, Type body, WorldPlaces place){  
 super(name, body, place);  
 setDieseases(Illness.*REVMATIT*);  
 addHistory(" своя история жизни и смерти из-за болезни");  
 }  
  
}

Illness:

package people;  
  
public enum Illness {  
 *REVMATIT*("Ревматит"),  
 *GRIPP*("Грипп"),  
 *HEALTHY*("Здоровый"),  
 *VETRIANKA*("Ветрянка");  
 private String name;  
 Illness(String name){  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

Person:

package people;  
  
import Exceptions.\*;  
import interfaces.\*;  
import places.WorldPlaces;  
import java.util.ArrayList;  
import java.util.Objects;  
  
public class Person implements Connect, View {  
 private String name;  
 private Type body;  
 private Feelings feelings = Feelings.*NORMAL*;  
 private Dialog dialoger = Dialog.*LISTENER*;  
 private Illness dieseases = Illness.*HEALTHY*;  
 private Action act = Action.*NOTHING*;  
 protected WorldPlaces place;  
 private ArrayList<String> histories = new ArrayList<String>();  
  
 public Person(String name, Type body) {  
 this.name = name;  
 this.body = body;  
 this.histories.add(" своя история жизни");  
 }  
 public Person(String name, Type body, WorldPlaces place){  
 this.name = name;  
 this.body = body;  
 this.histories.add(" своя история жизни");  
 this.place = place;  
 try {  
 if (place == null) {  
 throw new BadPlaceException();  
 }  
 }catch (BadPlaceException e){  
 throw e;  
 }  
  
 place.addContains(this);  
 }  
 public Person(String name, Type body, WorldPlaces place, Action act){  
 this.name = name;  
 this.body = body;  
 this.act = act;  
 this.histories.add(" своя история жизни");  
 this.place = place;  
 try {  
 if (place == null) {  
 throw new BadPlaceException();  
 }  
 }catch (BadPlaceException e){  
 throw e;  
 }  
 place.addContains(this);  
 }  
  
  
 private class Organism{  
 public void heal() {  
 if(!dieseases.equals(Illness.*HEALTHY*)) System.*out*.println(getName() + " когда то лечился от болезни " + getDieseases().getName() + " так как заболел ей после охоты, но в итоге умер.");  
 else System.*out*.println(getName() + " полностью здоров.");  
 }  
 }  
  
 public void changePlace(Person friend, WorldPlaces newPlace){  
 System.*out*.print(friend.getName()+" и "+getName()+" дошли до "+newPlace.getName()+". ");  
 this.place = newPlace;  
 friend.place= newPlace;  
 startWalkTo(friend, newPlace);  
 }  
 public void LookAround() {  
 System.*out*.println(getName() + " оглянулся вокруг и увидел: ");  
 for (int i = 0; i < place.getContains().size(); i++) {  
 if (place.getContains().get(i).equals(this)) {  
 } else {  
 System.*out*.println(place.getContains().get(i).getName()+", который "+place.getContains().get(i).getAct().getName());  
 }  
 }  
 for (int i = 0; i < place.getContains().size(); i++) {  
 if (place.getContains().get(i).equals(this)) {  
 } else {  
 startWalkTo(place.getContains().get(i),place);  
  
 break;  
 }  
 }  
 }  
 public void startWalkTo(Person friend, WorldPlaces place) {  
 place.addFootPrints(" немного следов " + getName());  
 place.addFootPrints(" немного следов " + friend.getName());  
 place.addContains(this);  
 place.addContains(friend);  
 System.*out*.println(getName() + " c " + friend.getName()+ " вместе начинают гулять по " + place.getName());  
 this.act=Action.*WALK*;  
 }  
 public void walkWithTo(Person friend, int time){  
 try {  
 try {  
 if (place == null) throw new BadPlaceException();  
 }catch (BadPlaceException e){  
 System.*out*.println(e.toString());  
 throw e;  
 }  
  
 if (time == 1) {  
 place.addFootPrints("немного следов " + getName());  
 place.addFootPrints("немного следов " + friend.getName());  
 System.*out*.println(getName() + " идёт по " + place.getName() + " с " + friend.getName() + " приблизительно " + time + " ч.");  
 } else if((time > 1) && (time < 5)) {  
 place.addFootPrints("немало следов " + getName());  
 place.addFootPrints("немало следов " + friend.getName());  
 System.*out*.println(getName() + " идёт по " + place.getName() + " с " + friend.getName() + " приблизительно " + time + " ч.");  
 } else if ((time > 4) && (time < 25)) {  
 place.addFootPrints("очень много следов " + getName());  
 place.addFootPrints("очень много следов " + friend.getName());  
 System.*out*.println(getName() + " идёт по " + place.getName() + " с " + friend.getName() + " приблизительно " + time + " ч. Тут "+place.getFootPrints().toString());  
 }  
 else{  
 throw new MegaLongTimeException();  
 }  
 }  
 catch (MegaLongTimeException e) {  
 System.*out*.println(e.toString());  
 System.*err*.println("ПЕРЕПРОВЕРЬТЕ ПРАВИЛЬНОСТЬ ВВЕДЁННОГО ВРЕМЕНИ ПРОГУЛКИ У "+this.name);  
 }  
 this.act=Action.*WALK*;  
 }  
 public void think(Person about, Boolean see){  
 class Thoughts{  
 public void thinks(){  
 System.*out*.println(getName() + " думает как же может выглядеть " + about.getName()+", и вдруг они как раз охотятся на него.");  
 }  
 public void remembers(){  
 System.*out*.println(getName() + " вспоминает " + about.getName()+" и как бы он поступил на его месте");  
 }  
 public void nosyalgates(){  
 System.*out*.println(getName() + " жалеет, что " + about.getName()+" сейчас не с ним, а в неизвестном месте.");  
 }  
 }  
 if(see) {  
 if(about.getDieseases()==Illness.*HEALTHY*){  
 new Thoughts().remembers();  
 }  
 else{new Thoughts().nosyalgates();}  
 }  
 else{new Thoughts().thinks();}  
 about.setFeelings(Feelings.*JOY*);  
 }  
 public void connectInDialogAboutPerson(Dialog type, Person friend, Person about) {  
 this.dialoger = type;  
 System.*out*.print(getName() + " заговорил с " + friend.getName());  
 class Topic{  
 void aboutPeople(Person about){  
 if ((about.getDieseases().equals(Illness.*HEALTHY*))) {  
 System.*out*.println(" о cледах " + about.getName()+", которые были в том месте, где они гуляют");  
 addHistory(" история об следах " + about.getName());  
 friend.addHistory(" история об следах " + about.getName() + ", которую рассказал " + getName());  
 } else {  
 System.*out*.println(" о болезне " + about.getName() + ", которая называется " + about.getDieseases().getName());  
 addHistory(" история об болезне " + about.getName());  
 friend.addHistory(" история об болезне " + about.getName() + ", которую рассказал " + getName());  
 about.DescribeTheOrganism();  
 }  
 }  
 }  
 new Topic().aboutPeople(about);  
 System.*out*.print("Список историй " + friend.getName() + " теперь состоит из:");  
 for (int i = 0; i < friend.getHistories().size(); i++) {  
 if (i != friend.getHistories().size() - 1) {  
 System.*out*.print(friend.getHistories().get(i) + ";");  
 } else {  
 System.*out*.println(friend.getHistories().get(i) + ".");  
 }  
 }  
 friend.think(about,false);  
  
 }  
 public void DescribeTheOrganism() {  
 Organism organism = new Organism();  
 organism.heal();  
 }  
  
  
  
 protected void addHistory(String elementToAdd) {  
 histories.add(elementToAdd);  
 }  
 public void setFeelings(Feelings feelings) {  
 this.feelings = feelings;  
 }  
 public Feelings getFeelings() {  
 return feelings;  
 }  
 public String getName() {  
 return name;  
 }  
 public Type getBody() {  
 return body;  
 }  
 public Dialog getDialoger() {  
 return dialoger;  
 }  
 public Illness getDieseases() {  
 return dieseases;  
 }  
 public void setDieseases(Illness dieseases) {  
 this.dieseases = dieseases;  
 }  
 public ArrayList<String> getHistories() {  
 return histories;  
 }  
 public Action getAct() {  
 return act;  
 }  
 public void setAct(Action act) {  
 this.act = act;  
 }  
 public WorldPlaces getPlace() {  
 return place;  
 }  
 public void setPlace(WorldPlaces place) {  
 this.place = place;  
 }  
  
  
  
 @Override  
 public void appear() {  
 System.*out*.println("Тип " + getName() + " - " + getBody().getName());  
 }  
 @Override  
 public void join() {  
 }  
 @Override  
 public void disconnect() {  
 }  
 @Override  
 public void request() {  
 }  
 @Override  
 public void outlook() {  
 }  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Person person = (Person) o;  
 return Objects.*equals*(name, person.name) && body == person.body && dialoger == person.dialoger && dieseases == person.dieseases;  
 }  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, body, dialoger, dieseases);  
 }  
 @Override  
 public String toString() {  
 return "Person{" +  
 "name='" + name + '\'' +  
 ", body=" + body +  
 ", dialoger=" + dialoger +  
 ", dieseases=" + dieseases +  
 '}';  
 }  
}

Type:

package people;  
  
public enum Type {  
 *ANIMAL*("Животное"),  
 *PEOPLE*("Человек"),  
 *SPIRIT*("Дух");  
 private String name;  
 Type(String name){  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

Package places

Forest:

package places;  
  
public class Forest extends WorldPlaces{  
 public Forest(String name) {  
 super(name);  
  
 }  
  
 public void birdsSinging(int value){  
 if(value==1)  
 System.*out*.println("В месте "+getName()+" погода - "+getWeather()+", поэтому слышутся птичьи голоса. ");  
 else  
 System.*out*.println("В месте "+getName()+" погода - "+getWeather()+", поэтому птички не хотят петь. ");  
 }  
 public void chanceRain(){  
 float ch = (float) (Math.*random*());  
 System.*out*.print("Шанс смены погоды составляет: " + ch+". ");  
 if(ch>0.5) {  
 TypeWeather[] type = {TypeWeather.*SUNNY*, TypeWeather.*CLOUDY*, TypeWeather.*RAINY*, TypeWeather.*WINDY*, TypeWeather.*SNOWY*};  
 int chance = (int) (Math.*random*() \* ((type.length) - 1 - 0 + 1) + 0);  
 if (type[chance].equals(getWeather())) {  
 chanceRain();  
 }  
 else {  
 setWeather(type[chance]);  
 }  
 }  
 }  
  
  
 @Override  
 public void join(){  
 appear();  
 setWeather(TypeWeather.*SUNNY*);  
 chanceRain();  
 if(getWeather().equals(TypeWeather.*SUNNY*.getName())){  
 birdsSinging(1);  
 }  
 else{  
 birdsSinging(0);  
 }  
 System.*out*.print("Сейчас тут находится: ");  
 for (int i = 0; i < getContains().size(); i++) {  
 System.*out*.print(getContains().get(i).getName()+", который "+getContains().get(i).getAct().getName()+"; ");  
 }  
 System.*out*.println();  
 }  
 @Override  
 public void appear(){  
 System.*out*.print(getName()+" очень таинственно выглядит и вокруг стоят высокие деревья. ");  
 }  
 @Override  
 public void outlook() {  
 System.*out*.println(getFootPrints());  
 }  
 @Override  
 public void disconnect() {}  
 @Override  
 public void request() {}  
 @Override  
 public String toString(){  
 return ("МЕСТО:" + getName());  
 }  
 @Override  
 public int hashCode() {  
 return super.hashCode() + getName().hashCode();  
 }  
 @Override  
 public boolean equals(Object obj) {  
 if (obj == null || getClass() != obj.getClass()) {  
 return false;  
 }  
 return this.hashCode() == obj.hashCode();  
 }  
}

NearSweetHomePit:

package places;  
  
public class NearSweetHomePit extends WorldPlaces{  
 public NearSweetHomePit(String name) {  
 super(name);  
  
 }  
  
  
 @Override  
 public void join(){  
 appear();  
 setWeather(TypeWeather.*SNOWY*);  
 }  
 @Override  
 public void appear(){  
 System.*out*.print("Всё вокруг "+getName()+" чудесно выглядит. ");  
 };  
 @Override  
 public void disconnect() {  
  
 }  
 @Override  
 public void request() {  
  
 }  
 @Override  
 public void outlook() {}  
}

Paradise:

package places;  
  
//дедус попал сюда  
public class Paradise extends WorldPlaces {  
 public Paradise(String name) {  
 super(name);  
 }  
  
  
 @Override  
 public void setWeather(TypeWeather type) {  
 System.*out*.print("тут всегда тепло и светит ярко солнце. ");  
 }  
 @Override  
 public void appear() {  
 System.*out*.print(getName() + " выглядит как мечта, ");  
 }  
 @Override  
 public void join() {  
 appear();  
 setWeather(TypeWeather.*SUNNY*);  
 System.*out*.print("Сейчас тут находится: ");  
 for (int i = 0; i < getContains().size(); i++) {  
 System.*out*.print(getContains().get(i).getName() + " ");  
 }  
 System.*out*.println();  
 }  
 @Override  
 public void outlook() {  
 }  
 @Override  
 public void disconnect() {  
 }  
 @Override  
 public void request() {  
 }  
 @Override  
 public String toString() {  
 return ("место:" + getName());  
 }  
 @Override  
 public int hashCode() {  
 return super.hashCode() + getName().hashCode();  
 }  
 @Override  
 public boolean equals(Object obj) {  
 if (obj == null || getClass() != obj.getClass()) {  
 return false;  
 }  
 return this.hashCode() == obj.hashCode();  
 }  
}

TypeWeather:

package places;  
//тип погоды  
public enum TypeWeather {  
 *SUNNY*("солнечно"),  
 *RAINY*("дождливо"),  
 *CLOUDY*("пасмурно"),  
 *WINDY*("ветрено"),  
 *SNOWY*("снежно");  
  
 private String name;  
 TypeWeather(String name){  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

WorldPlaces:

package places;  
  
import interfaces.\*;  
import people.Feelings;  
import people.Person;  
  
import java.util.ArrayList;  
  
//общая характеристика мест  
public abstract class WorldPlaces implements Connect, View {  
 private String name;  
 private TypeWeather weather;  
 private ArrayList<String> FootPrints = new ArrayList<String>();  
 public ArrayList<Person> contains = new ArrayList<Person>();  
  
 public WorldPlaces(String name){this.name=name;}  
  
 public void thereWereFootPrints(){  
 for (int i = 0; i < getContains().size(); i++) {  
 getContains().get(i).setFeelings(Feelings.*AFRAID*);  
 }  
 System.*out*.println("В месте "+getName()+" появились неизветсные следы и все стали "+Feelings.*AFRAID*.getName()+"ы");  
 }  
  
  
  
 public String getName(){return this.name;}  
 public String getWeather(){return weather.getName();}  
 public void setWeather(TypeWeather type){  
 this.weather=type;  
 System.*out*.println("В месте "+getName()+" "+type.getName()+".");  
 }  
 public void addFootPrints(String elementToAdd) {  
 FootPrints.add(elementToAdd);  
 }  
 public void addContains(Person elementToAdd) {  
 contains.add(elementToAdd);  
 }  
 public ArrayList<String> getFootPrints() {  
 return FootPrints;  
 }  
 public ArrayList<Person> getContains() {  
 return contains;  
 }  
  
  
 @Override  
 public void join(){  
 System.*out*.println("Место "+getName()+" подключилось к сценке");  
 appear();  
 setWeather(TypeWeather.*SUNNY*);  
 }  
 @Override  
 public void appear(){  
 System.*out*.println(getName()+" очень таинственно выглядит");  
 }  
}

src

Main:

public class Main {  
 public static void main(String[] args){  
 Story2 story = new Story2();  
 story.go();  
 }  
}

Story2:

import other.KillerTree;  
import people.\*;  
import places.\*;  
  
import java.util.ArrayList;  
  
public class Story2 {  
 public void go(){  
 WorldPlaces rosha = new Forest("Роща");  
 KillerTree.Apple apple = new KillerTree.Apple();  
 WorldPlaces ray = new Paradise("Рай");  
 WorldPlaces nearTheDomPita = new NearSweetHomePit("дорожка около дома Пита");  
 Person Puka = new Person("Пука", Type.*ANIMAL*, rosha, Action.*WALK*);  
 Person Buka = new Person("Бука", Type.*ANIMAL*,rosha, Action.*WALK*);  
 Person pita = new Person("Пяточoк", Type.*ANIMAL*, nearTheDomPita, Action.*CLEAN\_SNOW*);  
 Person vini = new Person("Вини-Пух", Type.*ANIMAL*, nearTheDomPita, Action.*WALK*){  
 @Override  
 public void LookAround(){  
 place.thereWereFootPrints();  
 System.*out*.print(getName()+" наклонил голову вниз и увидел на дорожке cледы");  
 for (int i = 0; i < place.getFootPrints().size(); i++) {  
 if(place.getFootPrints().get(i).contains("Бука")){  
 System.*out*.println(". Oни были ужасны и похожи на следы бук.");  
 break;  
 }  
 }  
 }  
 };  
 Person ded\_inside = new Grandfather("Дед\_Великий\_Охотник", Type.*SPIRIT*, ray);  
 Person Cristofer = new Person("Кристофер-Робин", Type.*PEOPLE*);  
  
 nearTheDomPita.join();  
 pita.LookAround();  
 pita.changePlace(vini, rosha);  
 rosha.join();  
 Buka.walkWithTo(Puka, 23);  
 vini.LookAround();  
 Person threeNonamePerson = new Person("Три неизвестные персоны",Type.*SPIRIT*, rosha,Action.*SEARCH\_FOR\_PREY*){  
 private Person death;  
 @Override  
 public void think(Person about, Boolean see) {  
 about.setFeelings(Feelings.*AFRAID*);  
 if(see){System.*out*.println(  
 getName()+" Увидели жертву и ждут момент для убийства "+about.getName());  
 about.setFeelings(Feelings.*AFRAID*);  
 setFeelings(Feelings.*BAD*);  
 setAct(Action.*KILL*);  
 apple.fallToPerson(about);  
 apple.outlook();  
 }  
 else{System.*out*.println(getName()+" Вглядываются в даль, ища жертву под именем "+about.getName());}  
 }  
 @Override  
 public void LookAround(){  
 System.*out*.println(getName()+" вглядыватся в следы и чувствуюn запах: ");  
 ArrayList<Person> deathNote = new ArrayList<Person>();  
 for (int i = 0; i < place.getContains().size(); i++) {  
 if(!this.equals(place.getContains().get(i))) {  
 deathNote.add(place.getContains().get(i));  
 System.*out*.print(place.getContains().get(i).getName()+" "+place.getContains().get(i).getBody().getName()+" "+place.getContains().get(i).getDieseases().getName());  
 System.*out*.println();  
 }  
 }  
 this.death = deathNote.get((int) (Math.*random*() \* ((deathNote.size()) - 1 - 0 + 1) + 0));  
 System.*out*.println("Выбрал рандомную жертву: "+death.getName());  
 think(death,false);  
 }  
 };  
 threeNonamePerson.LookAround();  
 vini.connectInDialogAboutPerson(Dialog.*SPEAKER*, pita, Buka);  
 pita.connectInDialogAboutPerson(Dialog.*SPEAKER*,vini,ded\_inside);  
 vini.think(Cristofer, true);  
 pita.think(ded\_inside,true);  
 threeNonamePerson.think(pita,true);  
  
 System.*out*.println("the end");  
 }  
}

# Результат работы

# Всё вокруг дорожка около дома Пита чудесно выглядит. В месте дорожка около дома Пита снежно.

# Пяточoк оглянулся вокруг и увидел:

# Вини-Пух, который гуляет

# Пяточoк c Вини-Пух вместе начинают гулять по дорожка около дома Пита

# Вини-Пух и Пяточoк дошли до Роща. Пяточoк c Вини-Пух вместе начинают гулять по Роща

# Роща очень таинственно выглядит и вокруг стоят высокие деревья. В месте Роща солнечно.

# Шанс смены погоды составляет: 0.7135621. В месте Роща дождливо.

# В месте Роща погода - дождливо, поэтому птички не хотят петь.

# Сейчас тут находится: Пука, который гуляет; Бука, который гуляет; Пяточoк, который гуляет; Вини-Пух, который гуляет;

# Бука идёт по Роща с Пука приблизительно 23 ч. Тут [ немного следов Пяточoк, немного следов Вини-Пух, очень много следов Бука, очень много следов Пука]

# В месте Роща появились неизветсные следы и все стали встревожены

# Вини-Пух наклонил голову вниз и увидел на дорожке cледы. Oни были ужасны и похожи на следы бук.

# Три неизвестные персоны вглядыватся в следы и чувствуюn запах:

# Пука Животное Здоровый

# Бука Животное Здоровый

# Пяточoк Животное Здоровый

# Вини-Пух Животное Здоровый

# Выбрал рандомную жертву: Бука

# Три неизвестные персоны Вглядываются в даль, ища жертву под именем Бука

# Вини-Пух заговорил с Пяточoк о cледах Бука, которые были в том месте, где они гуляют

# Список историй Пяточoк теперь состоит из: своя история жизни; история об следах Бука, которую рассказал Вини-Пух.

# Пяточoк думает как же может выглядеть Бука, и вдруг они как раз охотятся на него.

# Пяточoк заговорил с Вини-Пух о болезне Дед\_Великий\_Охотник, которая называется Ревматит

# Дед\_Великий\_Охотник когда то лечился от болезни Ревматит так как заболел ей после охоты, но в итоге умер.

# Список историй Вини-Пух теперь состоит из: своя история жизни; история об следах Бука; история об болезне Дед\_Великий\_Охотник, которую рассказал Пяточoк.

# Вини-Пух думает как же может выглядеть Дед\_Великий\_Охотник, и вдруг они как раз охотятся на него.

# Вини-Пух вспоминает Кристофер-Робин и как бы он поступил на его месте

# Пяточoк жалеет, что Дед\_Великий\_Охотник сейчас не с ним, а в неизвестном месте.

# Три неизвестные персоны Увидели жертву и ждут момент для убийства Пяточoк

# Дерево трещит и яблоко начинает быстро падать на Пяточoк и убивает его на смерть!

# Падающие яблоки делали все эти неизвестные следы, это самые жуткие убийцы в лесах. Их боялись даже буки

# the end

# Process finished with exit code 0

# Вывод

Во время выполнения данной лабораторной работы я научился применять принципы SOLID на практике, изучил понятия абстрактного класса, интерфейса и перечисления, а так же столкнулся с основными плюсом и минусом SOLID: очень сложно начать проект без предварительной разработки архитектуры и представления необходимых зависимостей, однако гораздо легче расширять и дополнять готовый проект, в котором используются принципы SOLID.