САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ.

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №3 ПО «ПРОГРАММИРОВАНИЮ»**

Вариант №3136

**Выполнила:**

Соколова Анастасия

Ефимова Катерина

Группа Р3101

**Преподаватель:**

Цопа Евгений Алексеевич

САНКТ-ПЕТЕРБУРГ, 2018

**Текст задания:**

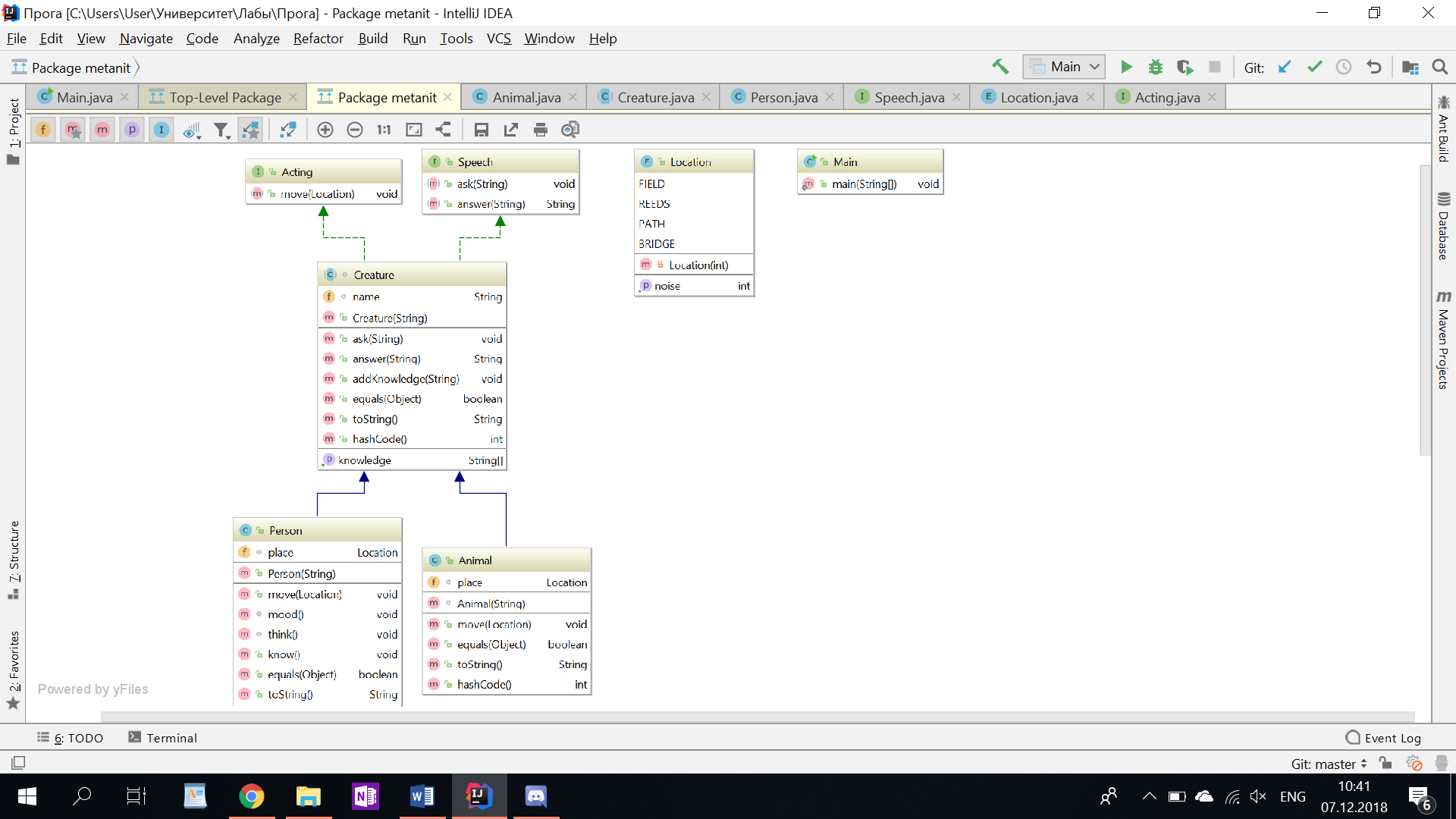
Прежде чем Иа успел ответить, раздался громкий треск, и из камышей появился Тигра собственной персоной. Кристофер Робин шел по лесной дорожке в таком солнечном и безоблачном настроении, как будто бы, например, дважды девятнадцать-- это пустяки, и думал о том, что, если он в такой день встанет на нижнюю перекладину перил моста и наклонится над рекой, он вдруг узнает все-все на свете и тогда он расскажет все это Пуху, который пока еще знает не все на свете. Но когда он подошел к мосту и увидел всех своих друзей, он понял, что сегодня совсем не такой день. а совершенно другой-- день, когда нужно что-то сделать.

**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.



**Исходный код**

public class Main {

public static void main(String[] args) {

Animal Pooh = new Animal("Винни");

Animal Ia = new Animal("Иа");

Animal Tiger = new Animal("Тигруля");

Pooh.move(Location.FIELD);

Pooh.ask("Как тебя зовут?");

System.out.println(Ia.answer("Меня зовут..."));

Tiger.move(Location.REEDS);

Person Chris = new Person("Кристофер Робин");

Chris.move(Location.PATH);

Chris.mood();

Chris.think();

Chris.move(Location.BRIDGE);

Chris.think();

Chris.know();

}

}

public interface Acting {  
 void move();  
}

public interface Speech {

void ask(String a);

String answer(String a);

}

import java.util.Arrays;

import java.util.Objects;

abstract class Creature implements Speech, Acting{

String name;

private String[] knowledge = new String[1];

public Creature(String name) {

this.name = name;

knowledge[0] = "Ничего не знает";

}

public void ask(String question) {

System.out.println(name + " спрашивает: \"" + question + "\"");

}

public String answer(String ans) {

return(name + " отвечает: \"" + ans + "\"");

}

public void addKnowledge(String knowledge) {

this.knowledge = Arrays.copyOf(this.knowledge, this.knowledge.length + 1);

this.knowledge[this.knowledge.length - 1] = knowledge;

}

public String[] getKnowledge() {

return knowledge;

}

@Override

public boolean equals(Object o) {

if (o == this)

return true;

if (o == null)

return false;

if (o.getClass() != this.getClass())

return false;

Creature creature = (Creature) o;

return this.name == creature.name;

}

@Override

public String toString() {

return "Creature name" + name;

}

@Override

public int hashCode() {

return Objects.hash(name);

}

}

import java.util.Objects;

public class Animal extends Creature {

Location place;

Animal(String name) {

super(name);

}

public void move(Location place) {

this.place = place;

switch (this.place) {

case FIELD: {

System.out.println(name + " находится на поле");

this.place.getNoise();

break;

}

case REEDS: {

System.out.println(name + " вышел из камышей в поле");

this.place.getNoise();

break;

}

case PATH: {

System.out.println(name + " пошел домой по дорожке");

this.place.getNoise();

break;

}

case BRIDGE: {

System.out.println(name + " стоит на мосту и ждет Кристофера");

this.place.getNoise();

break;

}

}

}

@Override

public boolean equals(Object o) {

if (o == this)

return true;

if (o == null)

return false;

if (o.getClass() != this.getClass())

return false;

Animal animal = (Animal) o;

return this.place == animal.place

&& name == animal.name;

}

@Override

public String toString() {

return "Animals name" + name + "place" + place;

}

@Override

public int hashCode() {

return Objects.hash(name, place);

}

}

import java.util.Objects;

public class Person extends Creature {

Location place;

public Person (String name) {

super(name);

}

@Override

public void move(Location place) {

this.place = place;

switch (this.place) {

case FIELD: {

System.out.println(name + "пришел к друзьям на поле");

break;

}

case REEDS: {

System.out.println(name + "полез в камыши искать Тигрулю");

break;

}

case PATH: {

System.out.println(name + " идет по дорожке");

break;

}

case BRIDGE: {

System.out.println(name + " подошел к мосту");

}

}

}

void mood() {

System.out.println("Настроение у него - солнечное и безоблачное");

}

void think() {

switch (place) {

case PATH: {

System.out.print("\"Если я встану на нижнюю перекладину перил моста и наклонюсь над рекой, ");

System.out.print("я узнаю все-все на свете и тогда я расскажу все это Пуху, который пока ");

System.out.println("еще знает не все на свете,\" - подумал " + name);

break;

}

case BRIDGE: {

if (getKnowledge().equals("Знает всё на свете")) {

System.out.println("Я теперь знаю всё-всё на свете, и я должен рассказать об этом Пуху!");

} else if ((place.getNoise() == 0 ) || (place.getNoise() == 1)) {

System.out.println("Сегодня день, когда нужно что-то сделать. - подумал " + name + ", увидев своих друзей.");

}

else {

System.out.println("Никаких планов на день. - подумал " + name);

}

break;

}

default: {

System.out.println("Никаких планов на день. - подумал " + name);

break;

}

}

}

public void know() {

addKnowledge(" знает, сколько будет дважды девятнадцать");

if (place.equals(Location.BRIDGE)) {

if ((place.getNoise() == 0 ) || (place.getNoise() == 1)) {

addKnowledge(" знает базовую математику");

} else {

addKnowledge(" знает все-все на свете");

}

}

System.out.println(name + getKnowledge()[getKnowledge().length - 1]);

}

@Override

public boolean equals(Object o) {

if (o == this)

return true;

if (o == null)

return false;

if (o.getClass() != this.getClass())

return false;

Animal animal = (Animal) o;

return this.place == animal.place

&& name == animal.name;

}

@Override

public String toString() {

return "Person name" + name + "place" + place;

}

@Override

public int hashCode() {

return Objects.hash(name, place);

}

}

public enum Location {

FIELD(0),

REEDS(1),

PATH(2),

BRIDGE(3);

private final int noise;

public int getNoise() {

return noise;

}

Location(int noise) {

this.noise = noise;

}

}

**Вывод:**

В ходе данной лабораторной работы мы изучили работу интерфейсов, абстрактных классов и перечисляемых типов. Лабораторная работа была выполнена в соответствии с принципами SOLID.