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:**

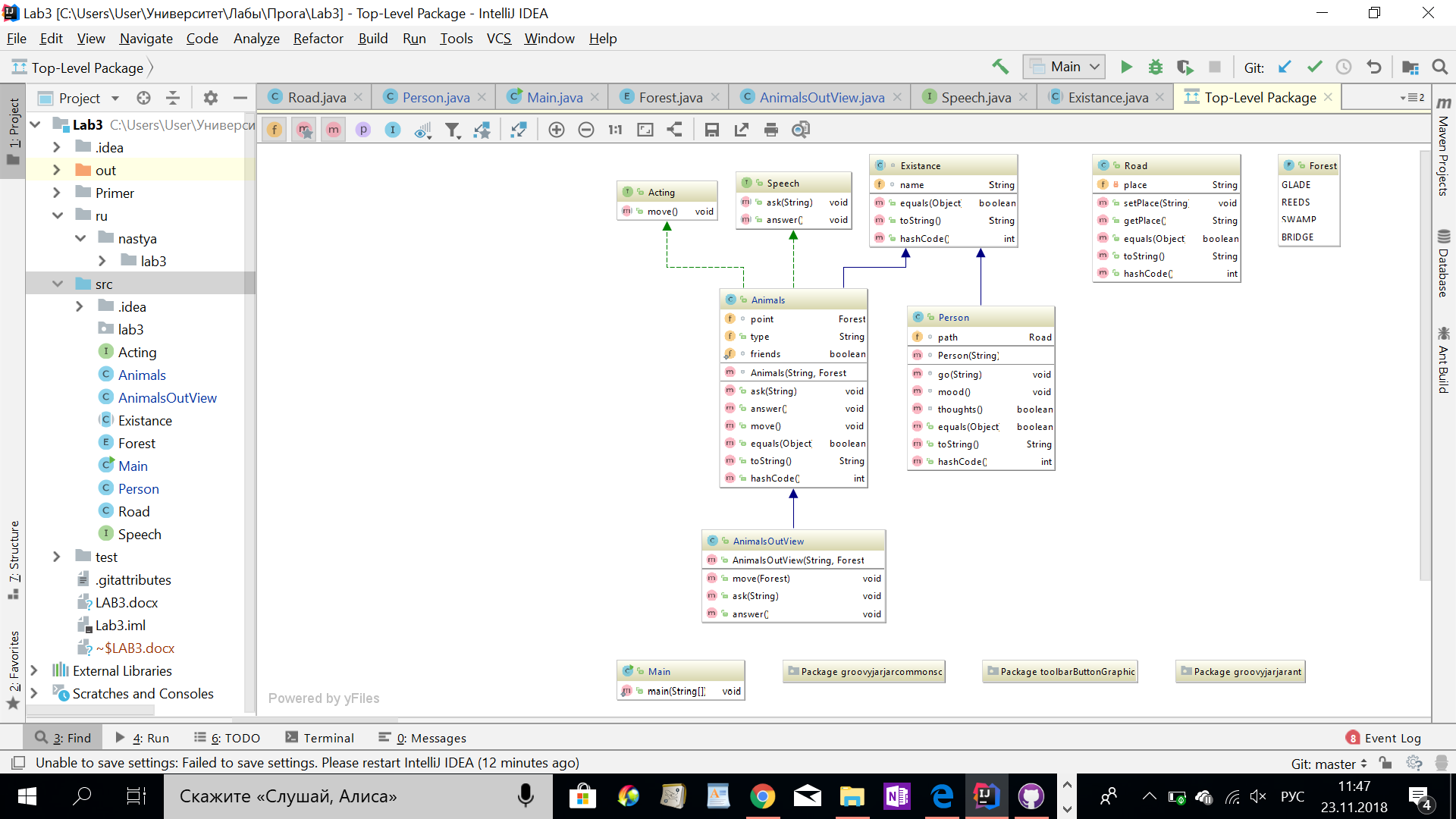
Прежде чем Иа успел ответить, раздался громкий треск, и из камышей появился Тигра собственной персоной. Кристофер Робин шел по лесной дорожке в таком солнечном и безоблачном настроении, как будто бы, например, дважды девятнадцать-- это пустяки, и думал о том, что, если он в такой день встанет на нижнюю 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) {  
 Animals pooh = new Animals("Винни", Forest.GLADE);  
 Animals ia = new Animals("Иа", Forest.GLADE);  
 AnimalsOutView Tiger = new AnimalsOutView("Тигруля", Forest.REEDS);  
 pooh.ask(ia.name);  
 ia.answer();  
 Tiger.move(Forest.GLADE);  
 Person Chris = new Person("Кристофер Робин");  
 Chris.go("Дорожка");  
 Chris.mood();  
 Chris.thoughts();  
 Chris.go("Мост");  
 Chris.thoughts();  
 }  
}

public interface Acting {  
 void move();  
}

public interface Speech {  
 void ask(String a);  
 void answer();  
}

abstract class Existance {  
 String name;  
 @Override  
 public boolean equals(Object object) {  
 return this == object;  
 }  
  
 @Override  
 public String toString() {  
 return "Existance [x = " + name + "]";  
 }  
  
 @Override  
 public int hashCode() {  
 return super.hashCode();  
 }  
}

public class Animals extends Existance implements Speech, Acting {  
 Forest point;  
 public String type;  
 static boolean *friends* = false;  
 //Animals() {  
  
 //}  
 Animals(String name, Forest area) {  
 super.name = name;  
 this.point = area;  
 switch (this.point) {  
 case *GLADE*:  
 this.type = "Поле";  
 this.*friends* = true;  
 break;  
 case *REEDS*:  
 this.type = "Камыши";  
 break;  
 case *SWAMP*:  
 this.type = "Болото";  
 break;  
 case *BRIDGE*:  
 this.type = "Мост";  
 break;  
 }  
 }  
 public void ask(String a) {  
 System.*out*.println(name + " спрашивает " + a);  
 }  
 public void answer() {  
 System.*out*.println(name + " отвечает");  
 }  
 public void move() {  
 System.*out*.println(name + "находится на месте: " + point);  
 }  
  
  
 @Override  
 public boolean equals(Object object) {  
 return this == object;  
 }  
  
 @Override  
 public String toString() {  
 return "Animals [x = " + name + "]";  
 }  
  
 @Override  
 public int hashCode() {  
 return super.hashCode();  
 }  
  
}

public class AnimalsOutView extends Animals /\*implements Acting\*/ {  
  
 public AnimalsOutView(String name, Forest area) {  
 super(name, area);  
 }  
 public void move(Forest point) {  
 System.*out*.println(name + " появился из места : " + type);  
 super.point = point;  
 }  
 public void ask(String a) {  
 }  
 public void answer() {  
 }  
}

public class Person extends Existance {  
 Road path = new Road();  
 Person(String name) **{** super.name = name;  
 **}** void go(String place) **{** path.setPlace(place);  
 System.*out*.println(name + " пошагал на место: " + path.getPlace());  
  
 **}** void mood() **{** System.*out*.println("Настроение : солнечное и безоблачное");  
 **}** boolean thoughts() **{** if (path.getPlace().equals("Дорожка")) {  
 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);  
 return true;  
 } else if (path.getPlace().equals("Мост") && (Animals.*friends*)) {  
 System.*out*.print("Сегодня день, когда нужно что-то сделать.");  
 System.*out*.println(" - подумал " + name + ", увидев своих друзей.");  
 return true;  
 } else {  
 System.*out*.println("никаких планов на день. - подумал " + name);  
 return false;  
 }  
 **}** @Override  
 public boolean equals(Object object) **{** return this == object;  
 **}** @Override  
 public String toString() **{** return "Person [x = " + name + "]";  
 **}** @Override  
 public int hashCode() **{** return super.hashCode();  
 **}**}

public enum Forest {  
 *GLADE*,  
 *REEDS*,  
 *SWAMP*,  
 *BRIDGE*}

public class Road {  
 private String place;  
 public void setPlace(String place) {  
 this.place = place;  
 }  
 public String getPlace() {  
 return place;  
 }  
 @Override  
 public boolean equals(Object object) {  
 return this == object;  
 }  
  
 @Override  
 public String toString() {  
 return "Forest [x = " + place + "]";  
 }  
  
 @Override  
 public int hashCode() {  
 return super.hashCode();  
 }

**Вывод:**

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