**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине

«ПРОГРАММИРОВАНИЕ»

Вариант № 88199

**Выполнил**:

Студент группы P3112

*Файзиев Фаридун Равшанович*

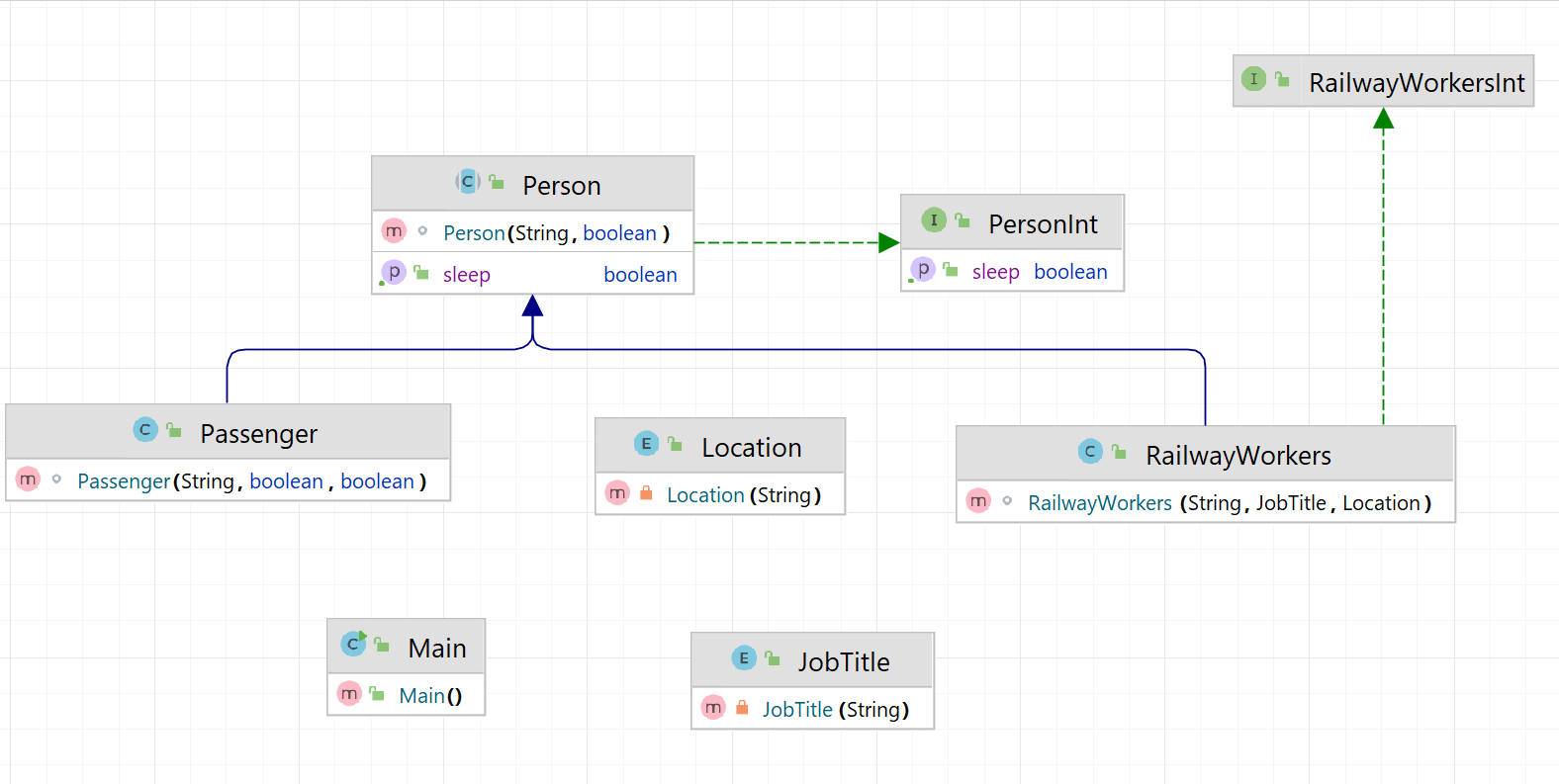
**Преподаватель**:

Гаврилов Антон Валерьевич

Санкт-Петербург, 2022 г.

Текст задания:

Мы оставили господина Скуперфильда как раз в тот момент, когда проводник высадил его из вагона в городе Паноптикуме. Некоторое время Скуперфильд стоял на перроне и осовело смотрел вслед удалявшемуся поезду. Как только поезд скрылся вдали, Скуперфильд подошел к стоявшей у края платформы лавочке и растянулся на ней, предварительно сунув под голову цилиндр и накрывшись газетой. Время было раннее. Еще было совсем темно, и никто не мешал Скуперфильду всхрапнуть. Вскоре наступил рассвет. На перроне появился какой-то железнодорожный начальник и, разбудив Скуперфильда, сказал, что спать здесь не полагается. В это время к станции подошел поезд. Перрон быстро заполнился сошедшими с поезда пассажирами. Встав с лавочки и напялив на голову цилиндр, Скуперфильд постоял в раздумье и отправился вслед за остальными пассажирами в город.

UML-диаграмма классов

Код программы:

*Class Main:*

public class Main {  
 public static void main(String[] args) {  
 Passenger Skuperfild = new Passenger("Скуперфильд", false, true);  
 RailwayWorkers Conductor = new RailwayWorkers("Проводник", JobTitle.*conductor*, Location.*IN\_TRAIN*);  
 RailwayWorkers Chief = new RailwayWorkers("Начальник", JobTitle.*railway\_chief*, Location.*ON\_PLATFORM*);  
  
 Conductor.check\_ticket(Skuperfild);  
 Chief.check\_person(Skuperfild);  
 Skuperfild.think("думает...");  
 Skuperfild.goTo("город");  
  
 System.*out*.println("\nИнформации " + Skuperfild.name + ":");  
 System.*out*.println("\t" + Skuperfild.toString() + " (toString)");  
 System.*out*.println("\tequals to Chief: " + Skuperfild.equals(Chief));  
 System.*out*.println("\thashCode: " + Skuperfild.hashCode());  
  
 }  
}

*Class Person:*

public abstract class Person implements PersonInt {  
 public String name;  
 public Location location;  
 private int energy = 100;  
 private int angry\_lvl = 0;  
 private boolean khrapit = false;  
  
 private boolean is\_sleep = false;  
  
 Person(String name, boolean khrapit){  
 this.name = name;  
 this.khrapit = khrapit;  
 }  
  
 public void look(String look){  
 System.out.println(this.name + " видит " + look);  
 }  
  
 public void say(String text, Person person){  
 System.out.println(this.name + " сказал " + person.name + " " + text);  
 }  
  
 public void hungry(int hungry){  
 System.*out*.println(this.name + " голоден!");  
 this.energy -= hungry;  
 if(this.energy < 25) {  
 sleep();  
 }  
 }  
  
 public void tired(int tired){  
 System.*out*.println(this.name + " устал!");  
 this.energy -= tired;  
 if(this.energy < 25) {  
 sleep();  
 }  
 }  
  
 public void angry(int angry\_lvl) {  
 this.angry\_lvl = this.angry\_lvl + angry\_lvl;  
 if(angry\_lvl > 0) {  
 System.*out*.println(this.name + " злой!");  
 this.energy -= 50;  
 }  
 }  
  
 public void sleep() {  
 if(!is\_sleep) {  
 System.out.println(this.name + " хочет спать...");  
 if(this.location == Location.IN\_TRAIN){  
 System.out.println(this.name + " уснул в купе");  
 }  
 else if(this.location == Location.ON\_PLATFORM){  
 this.location = Location.ON\_BENCH;  
 System.out.println(this.name + " уснул " + this.location.trans());  
 }  
  
 if(this.khrapit) {  
 System.out.println(this.name + " храпит )))");  
 }  
  
 this.is\_sleep = true;  
 }  
 }  
  
 public boolean isSleep() {  
 return this.is\_sleep;  
 }  
  
 public void toWake(Person person) {  
 System.out.println(this.name + " разбудил " + person.name);  
 say("спать здесь не полагается!", person);  
 person.toWakeUp();  
 }  
  
 public void toWakeUp() {  
 System.out.println(this.name + " проснулсся!");  
 this.is\_sleep = false;  
 this.energy += 50;  
 }  
  
 public void think(String text){  
 System.out.println(this.name + " " + text);  
 }  
  
 public void goTo(String text){  
 System.out.println(this.name + " идёт в " + text);  
 }  
  
 @Override  
 public String toString(){  
 return "имя: " + this.name;  
 }  
 @Override  
 public boolean equals(Object o){  
 if(this == o) {  
 return true;  
 }  
 if (o == null || getClass() != o.getClass()) {  
 return false;  
 }  
 Person person = (Person) o;  
  
 return name.equals(person.name) && location == person.location && energy == person.energy;  
  
 }  
 @Override  
 public int hashCode(){  
 int hash = 7;  
 hash = 31 \* hash + (name == null ? 0 : name.hashCode());  
 hash = 31 \* hash + (location == null ? 0 : location.hashCode());  
 return hash;  
 }  
}

*Class Passenger:*

public class Passenger extends Person{  
 public boolean ticket;  
 Passenger(String name, boolean ticket, boolean khrapit) {  
 super(name, khrapit);  
 super.location = Location.*IN\_TRAIN*;  
 this.ticket = ticket;  
 }  
}

*Class RailwayWorkers:*

public class RailwayWorkers extends Person implements RailwayWorkersInt{  
 private JobTitle job\_title;  
 RailwayWorkers(String name, JobTitle job\_title, Location location) {  
 super(name, false);  
 super.location = location;  
 this.job\_title = job\_title;  
 }  
  
 public void check\_ticket(Passenger passenger){  
 if(!passenger.ticket) {  
 passenger.location = Location.*ON\_PLATFORM*;  
 System.*out*.println(job\_title.trans() + " высадил из вагона " + passenger.name);  
 passenger.angry(50);  
 passenger.tired(15);  
 passenger.hungry(15);  
 }  
 }  
  
 public void check\_person(Person person){  
 if(person.location == Location.*ON\_BENCH* && person.isSleep()) {  
 super.toWake(person);  
 }  
 }  
}

*Enum JobTitle:*

public enum JobTitle {  
 *conductor*("Проводник"),  
 *railway\_chief*("Железнодорожный начальник");  
  
 private String translation;  
  
 JobTitle(String translation) {  
 this.translation = translation;  
 }  
  
 public String trans() {  
 return this.translation;  
 }  
}

*Enum Location:*

public enum Location {  
 *IN\_TRAIN*("в поезде"),  
 *ON\_PLATFORM*("на перроне"),  
 *ON\_BENCH*("на лавочке");  
  
 private String translation;  
  
 Location(String translation) {  
 this.translation = translation;  
 }  
  
 public String trans() {  
 return this.translation;  
 }  
  
  
}

*Interface PersonInt:*

public interface PersonInt {  
 void look(String look);  
 void say(String text, Person person);  
 void hungry(int hungry);  
 void tired(int tired);  
 void angry(int angry\_lvl);  
 void sleep();  
 boolean isSleep();  
 void toWake(Person person);  
 void toWakeUp();  
 void think(String text);  
 void goTo(String text);  
}

*Interface RailwayWorkersInt:*

public interface RailwayWorkersInt {  
 void check\_ticket(Passenger passenger);  
 void check\_person(Person person);  
}

*Результат:*

*Проводник высадил из вагона Скуперфильд*

*Скуперфильд злой!*

*Скуперфильд устал!*

*Скуперфильд голоден!*

*Скуперфильд хочет спать...*

*Скуперфильд уснул на лавочке*

*Скуперфильд храпит )))*

*Начальник разбудил Скуперфильд*

*Начальник сказал Скуперфильд спать здесь не полагается!*

*Скуперфильд проснулсся!*

*Скуперфильд думает...*

*Скуперфильд идёт в город*

*Информации Скуперфильд:*

*имя: Скуперфильд (toString)*

*equals to Chief: false*

*hashCode: -1907515097*

Выводы:

В процессе выполнения лабораторной работы я получил навыки использования

объектно-ориентированного подхода программирования при использовании языка Java.

Научился работать с интерфейсом, enum.