САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ

ИССЛЕДРОВАТЕЛЬСКИЙ

УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,

МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Программирование»

**Отчет**

По лабораторной работе №3

Вариант 290015

Студент

Федоров Евгений Константинович

Преподаватель

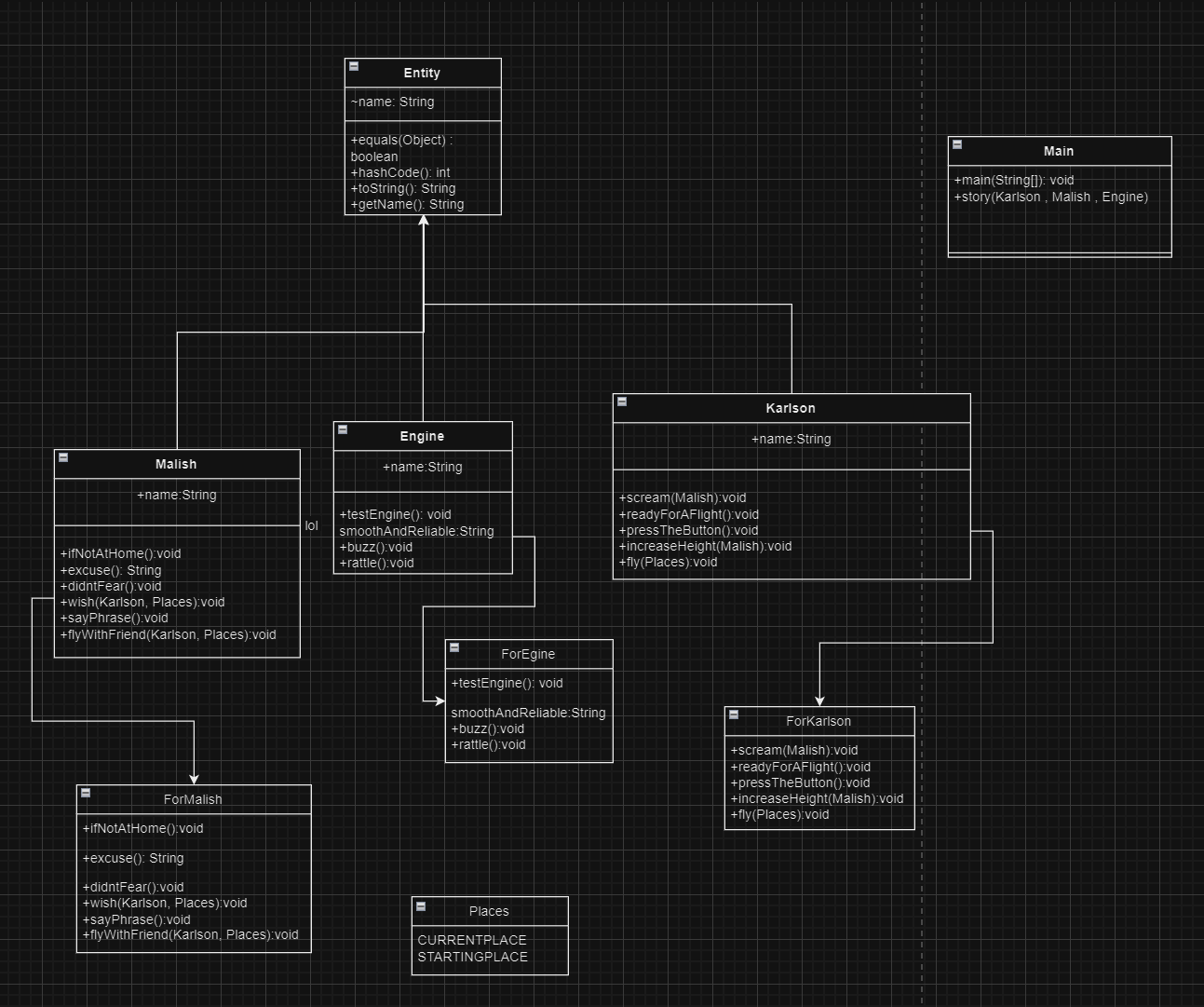
Наумова Надежда Александровна

Санкт-Петербург, 2023 г.

**Текст задания:**

“Потому, что я хотел ехать один”, - ответил Малыш.Вот и теперь то же самое. Он хочет отправиться с Карлсоном на крышу, поэтому лучше всего не просить разрешения. А если обнаружится, что его нет дома, он сможет оправдаться тем, что написал записку.Карлсон был готов к полёту. Он нажал кнопку на животе, и мотор загудел. - Залезай скорее мне на плечи, - крикнул Карлсон, - мы сейчас взлетим! И правда, они вылетели из окна и набрали высоту. Сперва Карлсон сделал небольшой круг над ближайшей крышей, чтобы испытать мотор. Мотор тарахтел так ровно и надёжно, что Малыш ни капельки не боялся.

**UML диаграмма:**



**Код программы:**

Package Entities:

Entity.java

package Entities;  
  
  
import java.util.Objects;  
  
public abstract class Entity{  
 protected String name;  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Entity entity = (Entity) o;  
 return Objects.*equals*(name, entity.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
  
 public Entity() {  
 }  
 public String getName() {  
 return this.name;  
 }  
  
 public String toString() {  
 return this.getName();  
 }  
  
}

Engine.java

package Entities;  
  
import Interfaces.ForEngine;  
  
public class Engine extends Entity implements ForEngine {  
  
 public Engine() {  
 name = "мотор";  
 }  
  
 @Override  
 public void testEngine() {  
 System.*out*.println("чтобы испытать " + super.name);  
 }  
  
  
 @Override  
 public void buzz() {  
 System.*out*.println(super.name + " загудел");  
  
 }  
  
 @Override  
 public void rattle() {  
 System.*out*.println(super.name + " тарахтел " + smoothAndReliable());  
 }  
}

Karlson.java

package Entities;  
  
import Enums.Places;  
import Interfaces.ForKarlson;  
  
;  
  
public class Karlson extends Entity implements ForKarlson{  
  
 public Karlson() {  
 name = "Карлсон";  
 }  
 @Override  
 public void readyForAFlight() {  
 System.*out*.println( super.name + " был готов к полету");  
 }  
  
  
  
  
  
 @Override  
 public void pressTheButton() {  
 System.*out*.println(super.name + " нажал на кнопку ");  
 }  
 @Override  
 public void scream(Malish m){  
 System.*out*.println("Залезай ко мне на плечи\" - крикнул " + super.name);  
 }  
 @Override  
 public void increaseHeight(Malish melkiy){  
 System.*out*.println("так что " +super.name + " и " + melkiy + " набрали высоту");  
 }  
  
  
@Override  
 public void fly(Places places) {  
 System.*out*.print(super.name + " сделал круг ");  
 if (places == Places.*CURRENTPLACE*){  
 System.*out*.print("над крышей ");  
 }  
  
 }  
  
}

Malish.java

package Entities;  
  
import Enums.Places;  
import Interfaces.ForMalish;  
  
  
public class Malish extends Entity implements ForMalish{  
  
 public Malish(){  
 name = "Малыш";  
 }  
  
 @Override  
 public void sayPhrase(){  
 System.*out*.print("\"Потому, что я хотел ехать один\"");  
  
  
 }  
 @Override  
 public void wish(Karlson karlson, Places places){  
 System.*out*.print("Он хочет отправиться с " + karlson + "ом");  
 if(places == Places.*CURRENTPLACE*);  
 {  
 System.*out*.println(" на крышу");  
 }  
 }  
  
  
 @Override  
 public void ifNotAtHome() {  
 System.*out*.println("Если " + this.name + "а " + "нет дома,");  
 }  
  
 @Override  
 public String excuse() {  
 return this.name + " оправдается тем, что у него есть записка.";  
 }  
  
 @Override  
 public void didntFear() {  
 System.*out*.println( "что " + this.name + " не боялся.");  
 }  
  
 @Override  
 public void flyWithFriend(Karlson karlson, Places places){  
 System.*out*.print(super.name + " вылетел вместе с " + karlson +"ом" );  
 if (places== Places.*STARTINGPLACE*){  
 System.*out*.println(" из окна");  
 }  
 }  
  
}

Package Enums:

Places:

package Enums;  
  
public enum Places {  
 *CURRENTPLACE*,  
 *STARTINGPLACE*}

Package Interfaces:

ForMalish:

package Interfaces;  
  
import Entities.Karlson;  
import Enums.Places;  
  
public interface ForMalish{  
 void ifNotAtHome();  
 String excuse();  
 void didntFear();  
 void wish(Karlson karlson, Places places);  
 void sayPhrase();  
 void flyWithFriend(Karlson karlson , Places places);  
}

ForKarlson:

package Interfaces;  
  
import Entities.Malish;  
import Enums.Places;  
  
public interface ForKarlson {  
  
 void scream(Malish m);  
  
 void readyForAFlight();  
  
 void pressTheButton();  
  
 void increaseHeight(Malish malish);  
 void fly(Places places);  
  
}

ForEngine:

package Interfaces;  
  
public interface ForEngine {  
 void testEngine();  
 default String smoothAndReliable(){  
 return "ровно и надежно";  
 }  
 void buzz();  
 void rattle();  
}

Main.java

import Entities.Engine;  
import Entities.Karlson;  
import Entities.Malish;  
import Enums.Places;  
public class Main {  
  
  
 public static void main(String... args) {  
 Malish melkiy = new Malish();  
 Karlson karlson = new Karlson();  
 Engine engine = new Engine();  
 *story*(melkiy, karlson, engine);  
 }  
 public static void story(Malish melkiy, Karlson karlson, Engine engine) {  
 melkiy.sayPhrase();  
 System.*out*.println("-ответил " + melkiy.getName());  
 System.*out*.println("Вот и теперь то же самое");  
 melkiy.wish(karlson, Places.*CURRENTPLACE*);  
 System.*out*.println("Поэтому лучше не просить разрешения");  
 melkiy.ifNotAtHome();  
 System.*out*.println(melkiy.excuse());  
 karlson.readyForAFlight();  
 karlson.pressTheButton();  
 engine.buzz();  
 karlson.scream(melkiy);  
 System.*out*.println("Мы сейчас взлетим!");  
 melkiy.flyWithFriend(karlson, Places.*STARTINGPLACE*);  
 karlson.increaseHeight(melkiy);  
 karlson.fly(Places.*CURRENTPLACE*);  
 engine.testEngine();  
 engine.rattle();  
 melkiy.didntFear();  
 }  
}

**Результат:**

"Потому, что я хотел ехать один"-ответил Малыш

Вот и теперь то же самое

Он хочет отправиться с Карлсоном на крышу

Поэтому лучше не просить разрешения

Если Малыша нет дома,

Малыш оправдается тем, что у него есть записка.

Карлсон был готов к полету

Карлсон нажал на кнопку

мотор загудел

Залезай ко мне на плечи" - крикнул Карлсон

Мы сейчас взлетим!

Малыш вылетел вместе с Карлсоном из окна

так что Карлсон и Малыш набрали высоту

Карлсон сделал круг над крышей чтобы испытать мотор

мотор тарахтел ровно и надежно

что Малыш не боялся.

**Вывод:**

Во время выполнения лабораторной работы я изучил принципы ООП SOLID; построил объектную модель по описанию предметной области, учитывая приципы SOLID; создал диаграмму UML для структуры проекта; до конца разобрался с интерфейсами и абстрактными классами, классом Object и его методами.