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 440

Студент

*Митрофанов Егор Юрьевич*

*P3114*

Преподаватель

*Письмак Алексей Евгеньевич*

Санкт-Петербург, 2019 г.

Текст задания:

*Малыш подошел к ней, чмокнул ее полную руку и сказал: Затем Малыш пошел к себе в комнату и стал ждать Карлсона. Они должны были сегодня вместе отправиться на крышу, и, если бы Карлсон был только выдумкой, как уверяет Кристер, вряд ли Малыш смог бы туда попасть.*

Программа должна удовлетворять следующим требованиям:

Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).

Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).

В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().

Программа должна содержать как минимум один перечисляемый тип (enum).

Порядок выполнения работы:

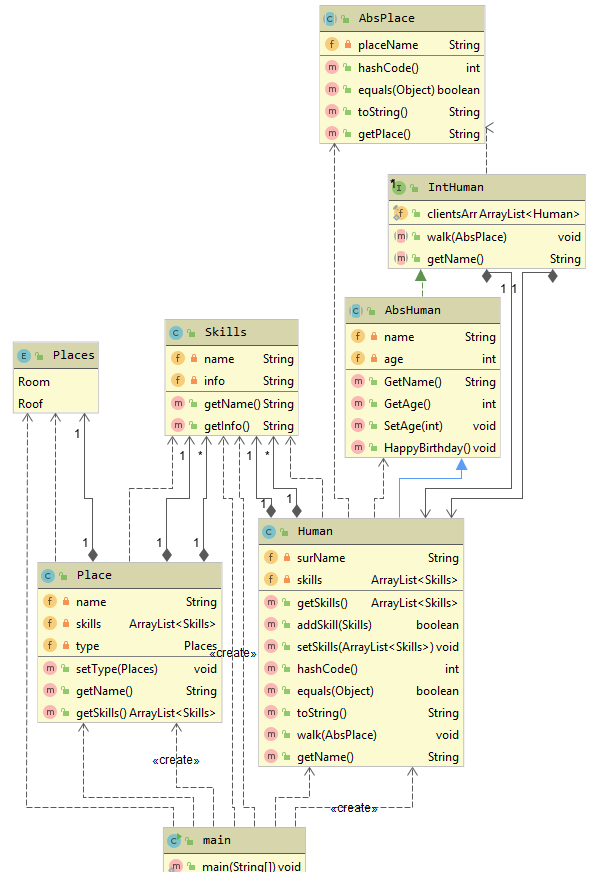
Доработать объектную модель приложения.

Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.

Согласовать с преподавателем изменения, внесённые в модель.

Модифицировать программу в соответствии с внесёнными в модель изменениями.

UML-диаграмма классов



Код программы:

**import** java.util.ArrayList;  
  
**public class** main {  
 **public static void** main(String[] args){  
 System.***out***.println();  
 ArrayList<Skills> skills = **new** ArrayList<>();  
 Human krister = **new** Human(**"Кристер"**);  
 krister.addSkill(**new** Skills(**"Доказывать"**, **"Карлсон не существует"**));  
 System.***out***.println();  
  
 Human karlson = **new** Human(**"Карлсон"**);  
 karlson.addSkill(**new** Skills(**"Летать"**, **"по крышам"**));  
 System.***out***.println();  
  
 Human malish = **new** Human(**"Малыш"**);  
 malish.addSkill(**new** Skills(**"Подойти"**, **null**));  
 malish.addSkill(**new** Skills(**"Чмокнуть"**, **"в полную руку"**));  
 System.***out***.println();  
  
 Place room = **new** Place(**"Локация1"**);  
 room.setType(Places.***Room***);  
 malish.addSkill(**new** Skills(**"Ждать"**, **"Карлсон"**));  
 System.***out***.println();  
  
 Place roof = **new** Place(**"Локация2"**);  
 room.setType(Places.***Roof***);  
 malish.addSkill(**new** Skills(**"Гулять"**, **"по крышам"**));  
 karlson.addSkill(**new** Skills(**"Гулять"**, **"по крышам"**));  
 }  
}

**public abstract class** AbsPlace {  
 **private** String **placeName**;  
 AbsPlace(String name) {  
 **placeName** = name;  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return super**.hashCode()+**this**.getPlace().hashCode();  
 }  
  
 @Override  
 **public boolean** equals(Object obj) {  
 **return** obj.hashCode() == **this**.hashCode();  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Место "** + **this**.getPlace();  
 }  
 **public** String getPlace() {  
 **return placeName**;  
 }  
}

**import** java.util.ArrayList;  
  
**public class** Place {  
 **private** String **name**;  
 **private** ArrayList<Skills> **skills** = **new** ArrayList<Skills>();  
  
 **private** Places **type**;  
  
 **public** Place(String \_name){  
 **name** = \_name;  
 System.***out***.println(**"Место - "** + **name** + **" успешно создано!"**);  
 }  
  
 **public void** setType(Places \_type){  
 **type** = \_type;  
 String typeName = **""**;  
 **switch** (\_type){  
 **case *Roof*** : typeName = **"Крыша"**;  
 **break**;  
 **case *Room*** : typeName = **"Комната"**;  
 **break**;  
 }  
 System.***out***.println(**name** +**" объявлена как "** + typeName);  
 }  
  
 **public** String getName() {  
 **return name** == **null** ? **"..."** : **name**;  
 }  
  
 **public** ArrayList<Skills> getSkills() {  
 **return skills**;  
 }  
  
}

**import** java.util.ArrayList;  
**public interface** IntHuman {  
 ArrayList<Human> ***clientsArr*** = **new** ArrayList<Human>();  
 **void** walk(AbsPlace place);  
 String getName();  
  
}

**public enum** Places {  
 ***Room***,  
 ***Roof***;  
}

**public abstract class** AbsHuman **implements** IntHuman {  
 **private** String **name**;  
 **private int age**;  
  
 **public** AbsHuman(){  
  
 }  
  
 **public** AbsHuman (String \_name){  
 **name** = \_name;  
  
 }  
  
 **public** AbsHuman (String \_name, **int** \_age){  
 **name** = \_name;  
 **age** = \_age;  
 }  
  
 **public** String GetName(){  
 **return name**;  
 }  
  
 **public int** GetAge() {  
 **return age**;  
 }  
  
 **public void** SetAge(**int** \_age) {  
 **age** = \_age;  
 }  
  
 **public void** HappyBirthday(){  
 **age**++;  
 }  
}

**public class** Skills {  
 **private** String **name**;  
 **private** String **info**;  
  
 **public** Skills(String \_name, String \_info){  
 **name** = \_name;  
 **info** = \_info;  
 }  
  
 **public** String getName(){  
 **return name**;  
 }  
  
 **public** String getInfo(){  
 **if** (**info** == **null**)  
 **return "..."**;  
 **else  
 return info**;  
  
 }  
  
}

**import** java.util.ArrayList;  
**public class** Human **extends** AbsHuman {  
 **private** String **surName**;  
 **private** ArrayList<Skills> **skills** = **new** ArrayList<Skills>();  
 **public** Human(){  
 System.***out***.println(**"Обычный человек успешно создан..."**);  
 }  
  
 **public** Human (String \_name){  
 **super** (\_name);  
 System.***out***.println(**"Человек - "** + \_name + **" успешно создан!"**);  
 }  
  
 **public** Human (String \_name, ArrayList<Skills> \_skills){  
 **super** (\_name);  
  
 **skills** = \_skills;  
 System.***out***.println(**"Человек - "** + \_name + **" успешно создан!"**);  
 }  
  
 **public** ArrayList<Skills> getSkills() {  
 **return skills**;  
 }  
  
 **pulic boolean** addSkill(Skills skill){  
 **if** (**skills**.add(skill)){  
 System.***out***.println(**"Объекту - \""** + **this**.GetName() + **"\" успешно присвоено умение "** + skill.getName()  
 + **" ("** + skill.getInfo()+ **")"**);  
 **return true**;  
 }  
 **else**{  
 System.***out***.println(**"При добавлении умения произошла ошибка..."**);  
 **return false**;  
 }  
 }  
 **public void** setSkills(ArrayList<Skills> skills) {  
 **this**.**skills** = skills;  
 }  
 @Override  
 **public int** hashCode() {  
 **return super**.hashCode()+**this**.getName().hashCode();  
 }  
 @Override  
 **public boolean** equals(Object obj) {  
 **return** obj.hashCode() == **this**.hashCode();  
 }  
 @Override  
 **public** String toString() {  
 **return "Человек по имени "** + **this**.getName();  
 }  
 @Override  
 **public void** walk(AbsPlace place) { }  
  
 @Override  
 **public** String getName() {  
 **return null**;  
 }  
}

Результат работы программы:

*Человек - Кристер успешно создан!*

*Объекту - "Кристер" успешно присвоено умение Доказывать (Карлсон не существует)*

*Человек - Карлсон успешно создан!*

*Объекту - "Карлсон" успешно присвоено умение Летать (по крышам)*

*Человек - Малыш успешно создан!*

*Объекту - "Малыш" успешно присвоено умение Подойти (...)*

*Объекту - "Малыш" успешно присвоено умение Чмокнуть (в полную руку)*

*Место - Локация1 успешно создано!*

*Локация1 объявлена как Комната*

*Объекту - "Малыш" успешно присвоено умение Ждать (Карлсон)*

*Место - Локация2 успешно создано!*

*Локация1 объявлена как Крыша*

*Объекту - "Малыш" успешно присвоено умение Гулять (по крышам)*

*Объекту - "Карлсон" успешно присвоено умение Гулять (по крышам)*

Вывод:

Во время выполнения лабораторной работы я научился работать с абстрактными классами, интерфейсами и нумераторами. Лучше освоил работу с полями, методами и классами и построение UML диаграмм.