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 №4

Вариант 440,142

Студент

*Митрофанов Егор Юрьевич*

*P3114*

Преподаватель

*Письмак Алексей Евгеньевич*

Санкт-Петербург, 2019 г.

Текст задания:

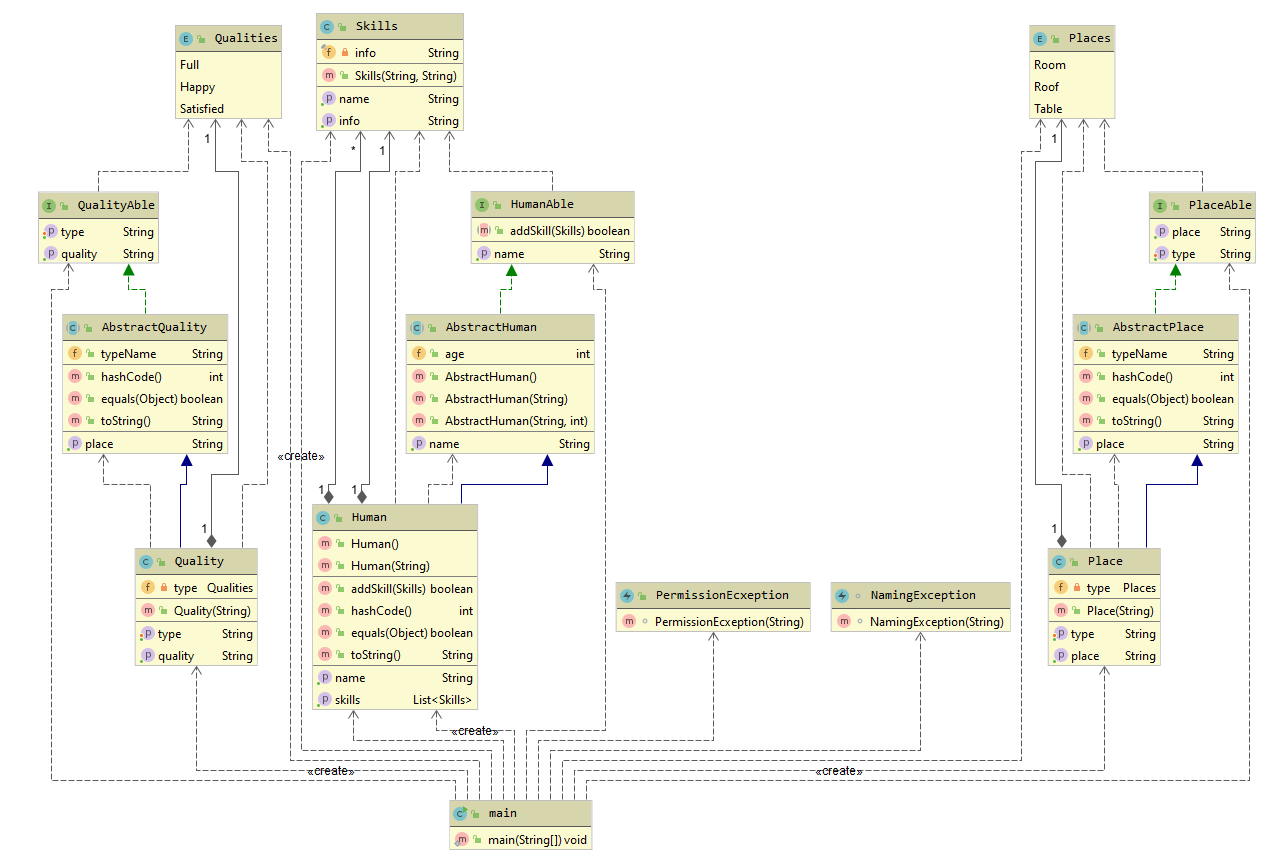
*Но вряд ли мама была бы так спокойна, если бы поняла до конца, что именно сказал ей Малыш. Ведь подумать только, где жил Карлсон! Малыш встал из-за стола сытый, веселый и вполне довольный жизнью. Шишка на лбу уже не болела, во рту был изумительный вкус плюшек с корицей, через кухонное окно светило солнце, и мама выглядела такой милой в своем клетчатом переднике. 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. "Я прилечу за тобой приблизительно часа в три, или в четыре, или в пять, но ни в коем случае не раньше шести", -- сказал ему Карлсон.*

Программа должна удовлетворять следующим требованиям:

1. *В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.*
2. *В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).*

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.*

UML-диаграмма классов

Код программы:

**public class** main {  
 **public static void** main(String[] args){  
 **try** {  
 StoryEnd.EndOfStory ending = **new** StoryEnd.EndOfStory();  
 System.***out***.println();  
 HumanAble karlson = **new** Human(**"Карлсон"**);  
 karlson.addSkill(**new** Skills(**"Летать"**, **null**));  
 System.***out***.println();  
  
 HumanAble mother = **new** Human(**"Мама"**);  
 karlson.addSkill(**new** Skills(**"Выглядеть мило"**, **null**));  
 karlson.addSkill(**new** Skills(**"Беспокоиться"**, **"за Малыша"**));  
 System.***out***.println();  
  
 HumanAble krister = **new** Human(**"Кристер"**);  
 krister.addSkill(**new** Skills(**"Доказывать"**, karlson.toString() + **" не существует"**));  
 System.***out***.println();  
  
 HumanAble malish = **new** Human(**"Малыш"**);  
 malish.addSkill(**new** Skills(**"Чмокнуть"**, **"в полную руку"**));  
  
 QualityAble quality1 = **new** Quality(**"Качество1"**);  
 QualityAble quality2 = **new** Quality(**"Качество2"**);  
 QualityAble quality3 = **new** Quality(**"Качество3"**);  
 quality1.setType(Qualities.***Full***);  
 quality2.setType(Qualities.***Happy***);  
 quality3.setType(Qualities.***Satisfied***);  
 System.***out***.println();  
  
 PlaceAble location1 = **new** Place(**"Локация1"**);  
 PlaceAble location2 = **new** Place(**"Локация2"**);  
 PlaceAble location3 = **new** Place(**"Локация3"**);  
 location1.setType(Places.***Table***);  
 location2.setType(Places.***Room***);  
 location3.setType(Places.***Roof***);  
  
 System.***out***.println();  
 malish.addSkill(**new** Skills(**"Ждать"**, karlson.toString()));  
 malish.addSkill(**new** Skills(**"Гулять"**, location3.getType()));  
 karlson.addSkill(**new** Skills(**"Гулять"**, location3.getType()));  
 System.***out***.println();  
  
 Human.Actions.*worry*(mother.getName(), karlson.getName(), location3.getType());  
 Human.Actions.*getup*(malish.getName(), quality1.getType(), quality2.getType(),  
 quality3.getType(), location1.getType());  
 Human.Actions.*description*(malish.getName(), mother.getName(), location2.getType());  
 Human.Actions.*kiss*(malish.getName(), mother.getName());  
 Human.Actions.*away*(malish.getName(), karlson.getName(), location2.getType());  
 Human.Actions.*walk*(malish.getName(), karlson.getName(), location3.getType());  
 Human.Actions.*think*(malish.getName(), krister.getName(), karlson.getName());  
 Human.Actions.*wait*(karlson.getName(), malish.getName());  
 ending.PrintInf();  
 }  
 **catch**(NamingException ex){  
 System.***out***.println(**"NamingException catched: "** + ex.getMessage());  
 }  
 **catch**(PermissionEcxeption ex){  
 System.***out***.println(**"PermissionException catched: "** + ex.getMessage());  
 }  
 }  
}

**public class** Human **extends** AbstractHuman {  
 **private final** List<Skills> **skills** = **new** ArrayList<Skills>();  
  
 **public** Human(){  
 System.***out***.println(**"Обычный человек успешно создан..."**);  
 }  
  
 **public** Human (String name){  
 **this**.**name** = name;  
 System.***out***.println(**"Человек - "** + name + **" успешно создан!"**);  
 }  
  
 **public** List<Skills> getSkills() {  
 **return skills**;  
 }  
  
 **public 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 static class** Actions{  
 **public static void** away(String name1, String name2, String place){  
 System.***out***.println(**"Затем "** + name1 + **" пошел в "** + place + **" ждать "** + name2);  
 }  
  
 **public static void** kiss(String name1, String name2){  
 System.***out***.println(name1 + **" подошел к "** + name2 + **" , поцеловал в полную руку и сказал: "** + **null** + **" !"**);  
 }  
  
 **public static void** walk(String name1, String name2, String place){  
 System.***out***.println(name1 + **" и "** + name2 + **" должны были идти на "** + place + **" вместе"**);  
 }  
  
 **public static void** think(String name1, String name2, String name3){  
 System.***out***.println(name1 + **" не сможет туда попасть, если "** + name2 + **" окажется прав насчет "** + name3);  
 }  
  
 **public static void** worry(String name1, String name2, String place){  
 System.***out***.println(name1 + **" вряд ли была спокойна, если бы поняла, что "** + name2 + **" живет на "** + place);  
 }  
  
 **public static void** getup(String name1, String quality1, String quality2, String quality3, String place){  
 System.***out***.println(**"Вставая из-за "** + place + **" , "** + name1 + **" чувствовал, что он "** + quality1 + **" "** +  
 quality2 +**" "** +quality3);  
 }  
  
 **public static void** description(String name1, String name2, String place){  
 System.***out***.println(name1 + **" чувствовал вкус плюшек, солнце светило в "** + place + **" ,и "** + name2 + **" вглядела милой"**);  
 }  
  
 **public static void** wait(String name1, String name2){  
 System.***out***.println(name1 + **" сказал, что прилетит за "** + name2 + **" не раньше шести..."**);  
 }  
 }  
  
  
  
 @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** String getName() {  
 **return name**;  
 }  
}

**public abstract class** AbstractHuman **implements** HumanAble {  
 **public** String **name**;  
 **public int age**;  
  
 **public** AbstractHuman(){  
 }  
  
 **public** AbstractHuman(String name){  
 **this**.**name** = name;  
 }  
  
 **public** AbstractHuman(String name, **int** age){  
 **this**.**name** = name;  
 **this**.**age** = age;  
 }  
  
 **public** String getName(){  
 **return name**;  
 }  
}

**public interface** HumanAble {  
 String getName();  
 **boolean** addSkill(Skills skills);  
}

**public class** Place **extends** AbstractPlace {  
 **private** Places **type**;  
  
 **public** Place(String name){  
 **this**.**placeName** = name;  
 System.***out***.println(**"Место - "** + name + **" успешно создано!"**);  
 }  
  
 **public void** setType(Places type){  
 **this**.**type** = type;  
 **typeName** = **""**;  
 **switch** (type){  
 **case *Roof*** : **typeName** = **"Крыша"**;  
 **break**;  
 **case *Room*** : **typeName** = **"Комната"**;  
 **break**;  
 **case *Table***: **typeName** = **"Стол"**;  
 **break**;  
 }  
 System.***out***.println(**placeName** +**" объявлена как "** + **typeName**);  
 }  
  
 **public** String getType() {  
 **return this**.**typeName**;  
 }  
  
 **public** String getPlace() {  
 **return placeName** == **null** ? **"..."** : **placeName**;  
 }  
}

**public abstract class** AbstractPlace **implements** PlaceAble {  
 **public** String **placeName**;  
 **public** String **typeName**;  
  
  
 @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**;  
 }  
}

**public interface** PlaceAble {  
 String getPlace();  
 **void** setType(Places type);  
 String getType();  
}

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

**public class** Quality **extends** AbstractQuality {  
 **private** Qualities **type**;  
  
 **public** Quality(String name){  
 **this**.**qualityName** = name;  
 System.***out***.println(**"Характеристика - "** + name + **" успешно создана!"**);  
 }  
  
 **public void** setType(Qualities type){  
 **this**.**type** = type;  
 **typeName** = **""**;  
 **switch** (type){  
 **case *Full*** : **typeName** = **"Сытый"**;  
 **break**;  
 **case *Satisfied***: **typeName** = **"Доволен жизнью"**;  
 **break**;  
 **case *Happy***: **typeName** = **"Веселый"**;  
 **break**;  
 }  
 System.***out***.println(**qualityName** +**" обозначено как "** + **typeName**);  
 }  
  
 **public** String getType() {  
 **return this**.**typeName**;  
 }  
  
 **public** String getQuality() {  
 **return qualityName** == **null** ? **"..."** : **qualityName**;  
 }  
}

**public abstract class** AbstractQuality **implements** QualityAble {  
 **public** String **qualityName**;  
 **public** String **typeName**;  
  
  
 @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 "Качество "** + **this**.getQuality();  
 }  
  
 **public** String getPlace() {  
 **return qualityName**;  
 }  
}

**public interface** QualityAble {  
 String getQuality();  
 **void** setType(Qualities type);  
 String getType();  
}

**public enum** Qualities {  
 ***Full***,  
 ***Happy***,  
 ***Satisfied***;  
}

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

**public class** StoryEnd {  
 **static class** EndOfStory {  
 **void** PrintInf() {  
 System.***out***.print(**" ..... To be continued ..... "**);  
 }  
 }  
}

**class** NamingException **extends** RuntimeException{  
  
 NamingException(String s){  
 **super**(s);  
 }  
}

**public class** PermissionEcxeption **extends** RuntimeException{  
 PermissionEcxeption(String message){  
 **super**((message.length()!=0) ? message : **"Undefined"**);  
 }  
}

Результат работы программы

Человек - Карлсон успешно создан!

Объекту - "Карлсон" успешно присвоено умение Летать (...)

Человек - Мама успешно создан!

Объекту - "Карлсон" успешно присвоено умение Выглядеть мило (...)

Объекту - "Карлсон" успешно присвоено умение Беспокоиться (за Малыша)

Человек - Кристер успешно создан!

Объекту - "Кристер" успешно присвоено умение Доказывать (Человек по имени Карлсон не существует)

Человек - Малыш успешно создан!

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

Характеристика - Качество1 успешно создана!

Характеристика - Качество2 успешно создана!

Характеристика - Качество3 успешно создана!

Качество1 обозначено как Сытый

Качество2 обозначено как Веселый

Качество3 обозначено как Доволен жизнью

Место - Локация1 успешно создано!

Место - Локация2 успешно создано!

Место - Локация3 успешно создано!

Локация1 объявлена как Стол

Локация2 объявлена как Комната

Локация3 объявлена как Крыша

Объекту - "Малыш" успешно присвоено умение Ждать (Человек по имени Карлсон)

Объекту - "Малыш" успешно присвоено умение Гулять (Крыша)

Объекту - "Карлсон" успешно присвоено умение Гулять (Крыша)

Мама вряд ли была спокойна, если бы поняла, что Карлсон живет на Крыша

Вставая из-за Стол , Малыш чувствовал, что он Сытый Веселый Доволен жизнью

Малыш чувствовал вкус плюшек, солнце светило в Комната ,и Мама вглядела милой

Малыш подошел к Мама , поцеловал в полную руку и сказал: null !

Затем Малыш пошел в Комната ждать Карлсон

Малыш и Карлсон должны были идти на Крыша вместе

Малыш не сможет туда попасть, если Кристер окажется прав насчет Карлсон

Карлсон сказал, что прилетит за Малыш не раньше шести...

..... To be continued .....

Вывод:

Во время выполнения лабораторной работы я научился работать с обработкой ошибок и исключений. Изучил локальные, абстрактные и внутренние классы. Лучше освоил работу с полями, методами и классами и построение UML диаграмм.