Университет ИТМО

Направление СППО

Лабораторная работа №4 по Программированию

Преподаватель: Горбунов Михаил Витальевич

Выполнил: Потапова Вера Сергеевна
Группа: P3110
Вариант: 10401.15

Санкт-Петербург
2020

Задание:

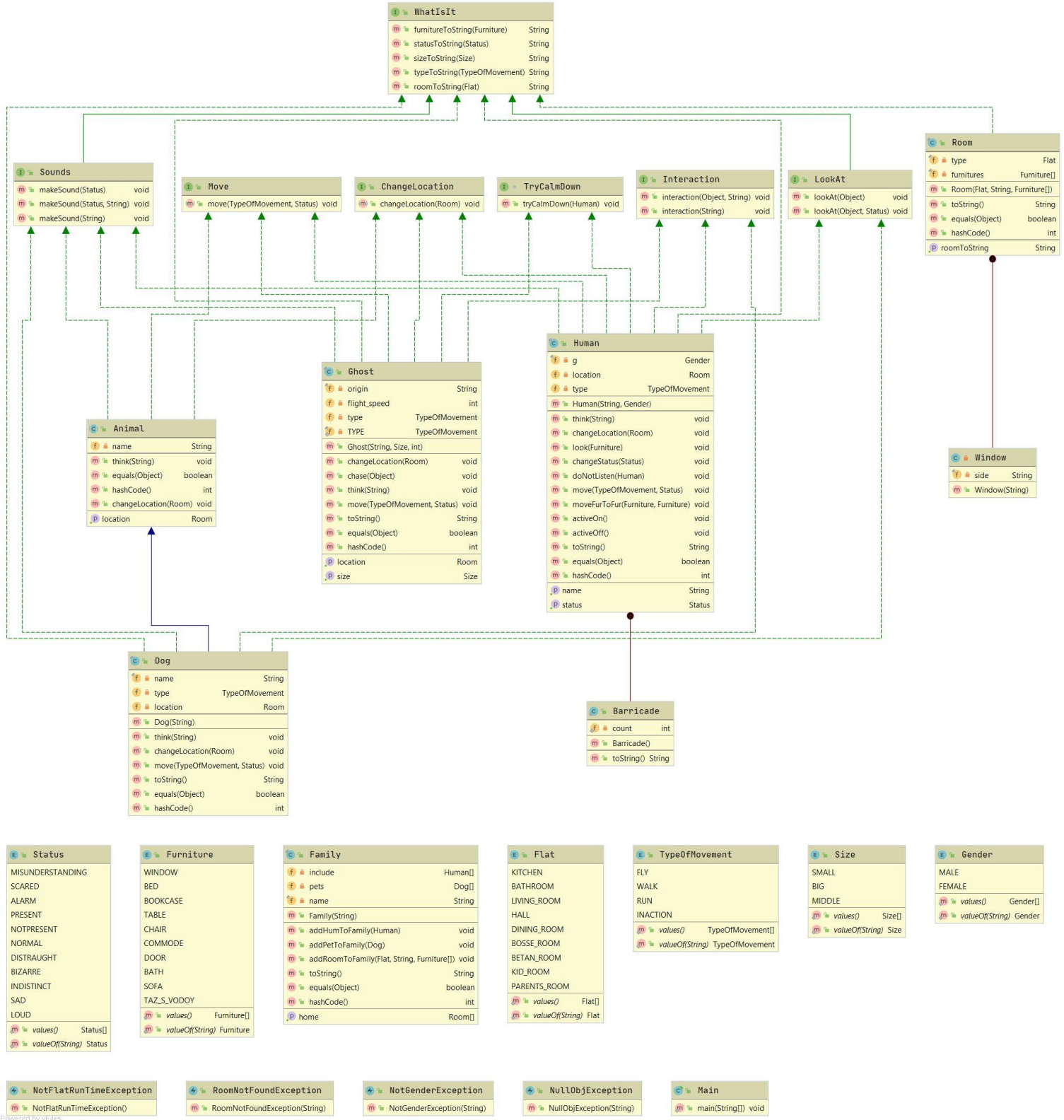
Доработать программу из лабораторной работы №3, обновив реализацию объектной модели в соответствии с новой версией описания предметной области.

Текст:

Малыш попытался ее успокоить. Однако фрекен Бок не слушала Малыша. Ее обезумевший взгляд был прикован к окну -- она следила за причудливым полетом привидения. Но маленькое привидение из Вазастана нельзя было убрать. Оно кружило в ночи, удалялось, вновь приближалось, то взмывая ввысь, то спускаясь пониже, и время от времени делало в воздухе небольшой кульбит. А печальные звуки не смолкали ни на мгновение. "Маленькое белое привидение, темное звездное небо, печальная музыка -- до чего все это красиво и интересно!" -- думал Малыш. Но фрекен Бок так не считала. Она вцепилась в Малыша: В квартире семьи Свантесон было пять комнат, кухня, ванная и передняя. У Боссе, у Бетан и у Малыша были свои комнатки, мама и папа спали в спальне, а кроме того, была столовая, где они собирались все вместе. Теперь, когда мама и папа были в отъезде, фрекен Бок спала в их комнате. Окно ее выходило в сад, а окно комнаты Малыша -- на улицу. Малыш сопротивлялся: нельзя же допустить, что бы все сорвалось теперь, после такого удачного начала! Но фрекен Бок упрямо стояла на своем: Малыш никак не мог этого понять, он сел на папину кровать, поглядел на перепуганную фрекен Бок и покачал головой. Но сейчас фрекен Бок и слышать не хотела о Фриде. Она продолжала придвигать всю мебель к двери -- за комодом последовали стол, стулья и этажерка. Перед столом образовалась уже настоящая баррикада. Но тут из-под папиной кровати раздался глухой голос, в котором звучало еще больше удовлетворения: И маленькое привидение стремительно, со свистом вылетело из-под кровати. И привидение разразилось долгим глухим смехом. Но фрекен Бок было не до смеха. Она кинулась к двери и стала расшвыривать мебель. В мгновение ока разобрав баррикаду, она с громким криком выбежала в переднюю. Привидение полетело следом, а Малыш побежал за ним. Последним мчался Бимбо и заливисто лаял. Он узнал привидение по запаху и думал, что началась веселая игра. Привидение, впрочем, тоже так думало. Но потом оно немного поотстало, чтобы получилась настоящая погоня. Так они носились по всей квартире -- впереди скакала фрекен Бок, а за ней мчалось привидение: в кухню и из кухни, в столовую и из столовой, в комнату Малыша и из комнаты Малыша и снова в кухню, большую комнату, комнату Малыша и снова, и снова... Фрекен Бок все время вопила так, что в конце концов привидение даже попыталось ее успокоить: Но все эти утешения не возымели никакого действия. Фрекен Бок продолжала голосить и метаться по кухне. А там все еще стоял на полу таз с водой, в котором она мыла ноги. Привидение преследовало ее по пятам. "Гей, гей", -- так и звенело в ушах; в конце концов фрекен Бок споткнулась о таз и с грохотом упала. При этом она издала вопль, похожий на вой сирены, но тут привидение просто возмутилось:

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).

Диаграмма:


WhatIsIt
furnitureToString(Furniture) String
statusToString(Status) String
sizeToString(Size) String
typeToString(TypeOfMovement) String
roomToString(Flat) String
Sounds
makeSound(Status) void
makeSound(Status, String) void
makeSound(String) void
Move
move(TypeOfMovement, Status) void
ChangeLocation
changeLocation(Room) void
TryCalmDown
tryCalmDown(Human) void
Interaction
interaction(Object, String) void
interaction(String) void
LookAt
lookAt(Object) void
lookAt(Object, Status) void
Room
type Flat
furnitures Furniture[]
Room(Flat, String, Furniture[])
toString() String
equals(Object) boolean
hashCode() int
roomToString String
Animal
name String
think(String) void
equals(Object) boolean
hashCode() int
changeLocation(Room) void
location Room
Ghost
origin String
flight_speed int
type TypeOfMovement
TYPE TypeOfMovement
Ghost(String, Size, int)
changeLocation(Room) void
chase(Object) void
think(String) void
move(TypeOfMovement, Status) void
toString() String
equals(Object) boolean
hashCode() int
location Room
size Size
Human
g Gender
location Room
type TypeOfMovement
Human(String, Gender)
think(String) void
changeLocation(Room) void
look(Furniture) void
changeStatus(Status) void
doNotListen(Human) void
move(TypeOfMovement, Status) void
moveFurToFur(Furniture, Furniture) void
activeOn() void
activeOff() void
toString() String
equals(Object) boolean
hashCode() int
name String
status Status
Window
side String
Window(String)
Dog
name String
type TypeOfMovement
location Room
Dog(String)
think(String) void
changeLocation(Room) void
move(TypeOfMovement, Status) void
toString() String
equals(Object) boolean
hashCode() int
Barricade
count int
Barricade()
toString() String
Status
MISUNDERSTANDING
SCARED
ALARM
PRESENT
NOTPRESENT
NORMAL
DISTRAUGHT
BIZARRE
INDISTINCT
SAD
LOUD
values() Status[]
valueOf(String) Status
Furniture
WINDOW
BED
BOOKCASE
TABLE
CHAIR
COMMODE
DOOR
BATH
SOFA
TAZ_S_VODOY
values() Furniture[]
valueOf(String) Furniture
Family
include Human[]
pets Dog[]
name String
Family(String)
addHumToFamily(Human) void
addPetToFamily(Dog) void
addRoomToFamily(Flat, String, Furniture[]) void
toString() String
equals(Object) boolean
hashCode() int
home Room[]
Flat
KITCHEN
BATHROOM
LIVING_ROOM
HALL
DINING_ROOM
BOSSE_ROOM
BETAN_ROOM
KID_ROOM
PARENTS_ROOM
values() Flat[]
valueOf(String) Flat
TypeOfMovement
FLY
WALK
RUN
INACTION
values() TypeOfMovement[]
valueOf(String) TypeOfMovement
Size
SMALL
BIG
MIDDLE
values() Size[]
valueOf(String) Size
Gender
MALE
FEMALE
values() Gender[]
valueOf(String) Gender
NotFlatRunTimeException
NotFlatRunTimeException()
RoomNotFoundException
RoomNotFoundException(String)
NotGenderException
NotGenderException(String)
NullObjException
NullObjException(String)
Main
main(String[]) void

Код программы:

Main.java

```
public class Main {
    public static void main(String[] args) {
        try {
            Family svanteson = new Family( name: "Svanteson");
            Human kid = new Human( name: "Malysh", Gender.MALE);
            Human frekenBok = new Human( name: "Freken Bok", Gender.FEMALE);
            Human mother = new Human( name: "mum", Gender.FEMALE);
            Human father = new Human( name: "dad", Gender.MALE);
            Human frida = new Human( name: "Frida", Gender.FEMALE);
            Human bosse = new Human( name: "Bosse", Gender.MALE);
            Human betan = new Human( name: "Betan", Gender.MALE);
            Dog bimbo = new Dog( name: "Bimbo");
            Ghost ghost = new Ghost( origin: "Vasastan", Size.SMALL, flight_speed: 100);
            svanteson.addHumToFamily(mother);
            svanteson.addHumToFamily(father);
            svanteson.addHumToFamily(bosse);
            svanteson.addHumToFamily(betan);
            svanteson.addHumToFamily(kid);
            svanteson.addHumToFamily(mother);
            svanteson.addPetToFamily(bimbo);
            svanteson.addRoomToFamily(Flat.BATHROOM, winSide: "NO", new Furniture[]{Furniture.DOOR, Furniture.BATH});
            svanteson.addRoomToFamily(Flat.BETAN_ROOM, winSide: "NO", new Furniture[]{Furniture.DOOR, Furniture.BED, Furniture.TABLE});
            svanteson.addRoomToFamily(Flat.BOSSE_ROOM, winSide: "at street", new Furniture[]{Furniture.DOOR, Furniture.BED, Furniture.TABLE});
            svanteson.addRoomToFamily(Flat.PARENTS_ROOM, winSide: "at garden", new Furniture[]{Furniture.DOOR, Furniture.BED, Furniture.TABLE, Furniture.CHAIR, Furniture.BOOKCASE, Furniture.COMMODE});
            svanteson.addRoomToFamily(Flat.HALL, winSide: "NO", new Furniture[]{Furniture.DOOR,Furniture.SOFA});
            svanteson.addRoomToFamily(Flat.KITCHEN, winSide: "at garden", new Furniture[]{Furniture.DOOR, Furniture.TAZ_S_VODOY});
            svanteson.addRoomToFamily(Flat.KID_ROOM, winSide: "at street", new Furniture[]{Furniture.DOOR, Furniture.TABLE, Furniture.BED});
            svanteson.addRoomToFamily(Flat.DINING_ROOM, winSide: "at street", new Furniture[]{Furniture.DOOR, Furniture.TABLE, Furniture.CHAIR});

            frekenBok.changeLocation(svanteson.getHome()[3]);
            kid.changeLocation(svanteson.getHome()[3]);
            ghost.changeLocation(svanteson.getHome()[3]);
            bimbo.changeLocation(svanteson.getHome()[3]);

            frekenBok.changeStatus(Status.ALARM);
            kid.tryCalmDown(frekenBok);
            frekenBok.doNotListen(kid);
            frekenBok.look(Furniture.WINDOW);
            frekenBok.lookAt(ghost);
            ghost.move(TypeOfMovement.FLY, Status.BIZARRE);
            frekenBok.lookAt(ghost);

            Sounds s = new Sounds() {
                @Override
                public void makeSound(Status st) {
                    System.out.println(statusToString(st) + " sounds did not subside for a moment.");
                }
            };
            s.makeSound(Status.SAD);
            kid.think( msg: "A little white ghost, a dark starry sky, sad music - how beautiful and interesting all this is!");
            frekenBok.interaction(kid, msg: " cling to ");
            kid.interaction( msg: "resisted:");
            kid.think( msg: "you can't allow everything to break down now, after such a good start!");
            frekenBok.interaction( msg: "stubbornly stood her ground.");
            kid.interaction(frekenBok, msg: "could not understand in any way.");
            kid.interaction(kid.furnitureToString(Furniture.BED), msg: "sat on");
            frekenBok.changeStatus(Status.SCARED);
            kid.lookAt(frekenBok, frekenBok.getStatus());
            kid.interaction( msg: "shook his head.");
            frekenBok.interaction(frida, msg: "did not want to hear about");
            frekenBok.interaction( obj: "the all furnitures", msg: "continued to move");
            frekenBok.moveFurToFur(Furniture.COMMODE, Furniture.DOOR);
            frekenBok.moveFurToFur(Furniture.TABLE, Furniture.DOOR);
            frekenBok.moveFurToFur(Furniture.CHAIR, Furniture.DOOR);
            frekenBok.moveFurToFur(Furniture.BOOKCASE, Furniture.DOOR);
            ghost.makeSound(Status.INDISTINCT);
            ghost.makeSound( msg: "sounded more satisfying");
            ghost.interaction(ghost.furnitureToString(Furniture.BED), msg: "swiftly, with a whistle flew out from under");
            ghost.interaction( msg: "burst into a long, dull laugh.");
            frekenBok.interaction( msg: "was not laughing.");
            frekenBok.interaction(frekenBok.furnitureToString(Furniture.DOOR), msg: "rushed to");
            frekenBok.interaction( msg: "began to throw furniture.");
            frekenBok.interaction( obj: "the Barricade", msg: "dismantled in the blink of an eye");
            frekenBok.activeOn();
            kid.activeOn();
            frekenBok.changeLocation(svanteson.getHome()[4]);
            frekenBok.interaction( msg: "with a loud cry, she ran into the hall.");
            ghost.changeLocation(svanteson.getHome()[4]);
            kid.changeLocation(svanteson.getHome()[4]);
            bimbo.changeLocation(svanteson.getHome()[4]);
            bimbo.makeSound( msg: "barked");
            bimbo.interaction(ghost, msg: "recognized by smell");
            bimbo.think( msg: "a fun game has begun!");
```

```
        ghost.think( msg: "a fun game has begun!");
        ghost.interaction( msg: "lagged behind a bit to make a real chase.");
        Interaction i = interaction(msg) -> { System.out.println("They " + msg); };
        i.interaction( msg: "ran all over the apartment.");
        frekenBok.changeLocation(svanteson.getHome()[5]);
        ghost.changeLocation(svanteson.getHome()[5]);
        frekenBok.changeLocation(svanteson.getHome()[7]);
        ghost.changeLocation(svanteson.getHome()[7]);
        frekenBok.changeLocation(svanteson.getHome()[6]);
        ghost.changeLocation(svanteson.getHome()[6]);
        frekenBok.changeLocation(svanteson.getHome()[5]);
        ghost.changeLocation(svanteson.getHome()[5]);
        frekenBok.changeLocation(svanteson.getHome()[4]);
        ghost.changeLocation(svanteson.getHome()[4]);
        frekenBok.changeLocation(svanteson.getHome()[6]);
        ghost.changeLocation(svanteson.getHome()[6]);
        frekenBok.makeSound( msg: "screamed");
        ghost.tryCalmDown(frekenBok);
        frekenBok.interaction( msg: "continued to scream and rush around the kitchen.");
        Interaction j = interaction(msg) -> {
                System.out.println(msg + frekenBok.furnitureToString(Furniture.TAZ_S_VODOY) + ".");
        };
        j.interaction( msg: "There still stood on the floor");
        ghost.chase(frekenBok);
        ghost.makeSound(Status.LOUD, msg: "Gey, gey!");
        frekenBok.interaction(frekenBok.furnitureToString(Furniture.TAZ_S_VODOY),  msg: "tripped over");
        frekenBok.interaction( msg: "fell with a crash.");
        frekenBok.makeSound(Status.LOUD, msg: "a scream like a siren");
    }catch (NotGenderException | RoomNotFoundException ex){...}
  }
}
```

Animal.java

```
import java.util.Objects;

public abstract class Animal implements Move, Sounds, ChangeLocation {
    private String name;
    private Room location;

    public void think(String msg) { System.out.println("'" + msg + "' - thought " + name); }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Animal animal = (Animal) o;
        return Objects.equals(name, animal.name) &&
                Objects.equals(location, animal.location);
    }

    @Override
    public int hashCode() { return Objects.hash(name, location); }

    @Override
    public void changeLocation(Room room) { this.location = room; }

    public Room getLocation() { return location; }
}
```

Dog.java

```java
import java.util.Objects;

public final class Dog extends Animal implements LookAt, Sounds, Interaction, WhatIsIt{
    private final String name;
    private TypeOfMovement type;
    public Dog(String name) {
       this.name = name;
       this.type = TypeOfMovement.RUN;
    }
    private Room location;

    public void think(String msg) { System.out.println("'" + msg + "' - thought " + toString()); }

    @Override
    public void changeLocation(Room room){
        if (location != null) {
            System.out.println(this.toString() + " " + typeToString(this.type) + " from " + location.getRoomToString() + " to " + room.getRoomToString());
        }
        this.location = room;
    }

    public void move(TypeOfMovement type, Status status) {
        this.type = type;
        System.out.println(toString() + " " + typeToString(type) + " " + statusToString(status) + ".");
    }

    @Override
    public String toString() { return this.name; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        if (!super.equals(o)) return false;
        Dog dog = (Dog) o;
        return type == dog.type &&
                Objects.equals(location, dog.location);
    }

    @Override
    public int hashCode() {
        return Objects.hash(super.hashCode(), type, location);
    }

}
```

Family.java

```java
import java.util.Arrays;
import java.util.Objects;

public final class Family {
    private Human[] include;
    private Dog[] pets;
    private final String name;
    private Room[] home;

    public Family(String name){
        this.name = name;
        this.include = new Human[0];
        this.pets = new Dog[0];
        this.home = new Room[0];
        System.out.println("Family " + name + " was created.");
    }

    public Room[] getHome(){
        return home;
    }

    public void addHumToFamily(Human human){
        Human[] include1 = new Human[include.length + 1];
        int i = 0;
        for (Human h : include){
            include1[i] = h;
            i ++;
        }
        include1[i] = human;
        this.include = include1;
    }

    public void addPetToFamily(Dog dog){
        Dog[] include1 = new Dog[pets.length + 1];
        int i = 0;
        for (Dog h : pets){
            include1[i] = h;
            i ++;
        }
        include1[i] = dog;
        this.pets = include1;
    }

    public void addRoomToFamily(Flat type, String winSide, Furniture[] fur){
        Room room = new Room(type, winSide, fur);
        Room[] include1 = new Room[home.length + 1];
        int i = 0;
        for (Room h : home){
            include1[i] = h;
            i ++;
        }
        include1[i] = room;
        this.home = include1;
    }

    @Override
    public String toString() {
        return this.name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Family family = (Family) o;
        return Arrays.equals(include, family.include) &&
                Arrays.equals(pets, family.pets) &&
                Objects.equals(name, family.name) &&
                Arrays.equals(home, family.home);
    }

    @Override
    public int hashCode() {
        int result = Objects.hash(name);
        result = 31 * result + Arrays.hashCode(include);
        result = 31 * result + Arrays.hashCode(pets);
        result = 31 * result + Arrays.hashCode(home);
        return result;
    }
}
```

Ghost.java

```java
import java.util.Objects;

public final class Ghost implements WhatIsIt, Move, Sounds, ChangeLocation, Interaction, TryCalmDown{
    private final String origin;
    private final Size size;
    private Room location;
    private int flight_speed;
    private TypeOfMovement type;
    private static final TypeOfMovement TYPE = TypeOfMovement.FLY;

    public Ghost (String origin, Size size, int flight_speed){
        this.location = null;
        this.origin = origin;
        this.size = size;
        this.flight_speed = flight_speed;
    }

    @Override
    public void changeLocation(Room room){
        if (location != null) {
            System.out.println(this.toString() + " " + typeToString(this.type) + " from " + location.getRoomToString() + " to " + room.getRoomToString());
        }
        this.location = room;
    }

    public void chase(Object obj){
        System.out.println(this.toString() + " chase " + obj.toString());

    }

    public void think(String msg) { System.out.println("'" + msg + "' - thought " + toString()); }

    public Room getLocation() { return location; }
```

```java
    @Override
    public void move(TypeOfMovement type, Status status) {
        this.type = type;
        System.out.println(toString() + " " + typeToString(type) + " " + statusToString(status));
        if (flight_speed > 90){
            System.out.println("It circled in the night, receded, approached again, now soaring up, then descending below, and from time to time did a little somersault in the air.");
        }
    }

    @Override
    public String toString() { return (sizeToString(getSize()) + " ghost from " + origin); }

    public Size getSize() { return size; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Ghost ghost = (Ghost) o;
        return flight_speed == ghost.flight_speed &&
                Objects.equals(origin, ghost.origin) &&
                size == ghost.size &&
                Objects.equals(location, ghost.location) &&
                type == ghost.type;
    }

    @Override
    public int hashCode() { return Objects.hash(origin, size, location, flight_speed, type); }
}
```

Human.java

```java
import java.util.Objects;

public final class Human implements TryCalmDown, WhatIsIt, Move, LookAt, Sounds, ChangeLocation, Interaction{
    private final String name;
    private final Gender g;
    private Room location;
    private TypeOfMovement type;
    private Status status;

    public Human(String name, Gender g) throws NotGenderException{
        this.name = name;
        this.location = null;
        if (g != Gender.FEMALE && g != Gender.MALE){
            throw new NotGenderException("Gender not found!");
        } else {
            this.g = g;
        }
        this.status = Status.NORMAL;
        this.type = TypeOfMovement.WALK;
    }

    public static class Barricade{
        private static int count = 0;

        public Barricade(){
            count++;
            if (count > 3){
                System.out.println("A real barricade has been created.");
            }
        }

        @Override
        public String toString() { return "the Barricade"; }
    }

    public void think(String msg) { System.out.println("'" + msg + "' - thought " + name); }

    @Override
    public void changeLocation(Room room) throws RoomNotFoundException{
        if (room == null) {
            throw new RoomNotFoundException("Warning: ROOM FOR MOVE NOT FOUND!");
        }else if (location != null){
            System.out.println(this.getName() + " " + typeToString(this.type) + " from " + location.getRoomToString() + " to " + room.getRoomToString());
            this.location = room;
        }else{
            this.location = room;
        }
    }

    public void look(Furniture furniture){
        class Sight implements WhatIsIt {
            private final Status status;

            public Sight () { this.status = Human.this.status; }

            @Override
            public String toString() { return "Sight"; }
        }
        Sight sight = new Sight();
        System.out.println(getName() + "'s " + sight.statusToString(status) + ' ' + sight.toString() + ' ' +  " were fixed on the "  + furnitureToString(furniture) + ".");
    }

    public void changeStatus(Status status){
        this.status = status;
        System.out.println(getName() + " change status to " + statusToString(getStatus()) + '.');

    }

    public void doNotListen(Human x) { System.out.println(getName() + " didn't listen to " + x.getName()); }

    public String getName(){ return name; }

    public Status getStatus() { return status; }
```

```
    public void move(TypeOfMovement type, Status status) {
        this.type = type;
        System.out.println(getName() + " " + typeToString(type) + " " + statusToString(status) + ".");
    }

    public void moveFurToFur(Furniture furniture1, Furniture furniture2){
        System.out.println(this.toString() + " move " + furnitureToString(furniture1) + " to " + furnitureToString(furniture2) + ".");
        if (furniture2 == Furniture.DOOR){
            new Barricade();
        }
    }

    public void activeOn() { this.type = TypeOfMovement.RUN; }

    public void activeOff() { this.type = TypeOfMovement.WALK; }

    @Override
    public String toString() { return getName(); }


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Human human = (Human) o;
        return Objects.equals(name, human.name) &&
                g == human.g &&
                Objects.equals(location, human.location) &&
                type == human.type &&
                status == human.status;
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, g, location, type, status);
    }
}
```

Room.java

```
import java.util.Arrays;
import java.util.Objects;

public final class Room implements WhatIsIt{
    private final Flat type;
    private final Furniture[] furnitures;

    private class Window {
        private final String side;
        public Window(String side){
            this.side = side;
            System.out.println("The window in the " + roomToString(type) + " looks out onto " + side + ".");
        }
    }

    public String getRoomToString() { return roomToString(type); }

    public Room(Flat type, String winSide, Furniture[] fur) throws NotFlatRunTimeException{
        if (!(type instanceof Flat) || type == null || !(winSide instanceof String) || !(fur instanceof Furniture[])){
            throw new NotFlatRunTimeException();
        }
        this.type = type;
        this.furnitures = fur;
        new Window(winSide);
    }

    @Override
    public String toString() { return roomToString(this.type); }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Room room = (Room) o;
        return type == room.type &&
                Arrays.equals(furnitures, room.furnitures);
    }
```

```
    @Override
    public int hashCode() {
        int result = Objects.hash(type);
        result = 31 * result + Arrays.hashCode(furnitures);
        return result;
    }
}
```

ChangeLocation.java

```
public interface ChangeLocation {
    void changeLocation (Room room) throws RoomNotFoundException;
}
```

Interaction.java

```
public interface Interaction {
    default void interaction(Object obj, String msg){
        System.out.println(this.toString() + " " + msg + " " + obj.toString() + ".");
    }

    default void interaction(String msg) { System.out.println(this.toString() + " " + msg); }
}
```

LookAt.java

```
public interface LookAt extends WhatIsIt{
    default void lookAt(Object obj) throws NullObjException{
        if (obj == null) throw new NullObjException("Warning: you try to look at null-object!");
        System.out.println(this.toString() + " follow at " + obj.toString());
    }

    default void lookAt(Object obj, Status status) throws NullObjException{
        if (obj == null) throw new NullObjException("Warning: you try to look at null-object!");
        System.out.println(this.toString() + " look at " + statusToString(status) + " " + obj.toString());
    }
}
```

Move.java

```
public interface Move {
    void move(TypeOfMovement type, Status status);
}
```

Sounds.java

```
public interface Sounds extends WhatIsIt{
    default void makeSound(Status st){
        System.out.println(statusToString(st) + " sounds.");
    }

    default void makeSound(Status st, String msg){
        System.out.println(this.toString() + " " + statusToString(st) + " says: '" + msg + "'.");
    }

    default void makeSound(String msg) { System.out.println(this.toString() + " says:' " + msg + "'."); }
}
```

TryCalmDown.java

```
interface TryCalmDown {
    default void tryCalmDown(Human x){
        System.out.println(this.toString() + " tried to calm down " + x.getName());
        if (Math.random()<0.1){
            x.changeStatus(Status.NORMAL);
            System.out.println(x.getName() + " calmed down a bit.");
            System.out.println("Further narration does not make sense.");
        }
        else {
            System.out.println(x.getName() + " could not calm down.");
        }
    }
}
```

WhatIsIt.java

```java
public interface WhatIsIt {
    default String furnitureToString(Furniture furniture) {
        switch (furniture) {
            case BED:
                return "bed";
            case TAZ_S_VODOY:
                return "a basin of water in which she washed her feet";
            case BATH:
                return "bath";
            case BOOKCASE:
                return "bookcase";
            case TABLE:
                return "table";
            case DOOR:
                return "door";
            case CHAIR:
                return "chair";
            case COMMODE:
                return "commode";
            default:
                return "";
        }
    }

    default String statusToString(Status status) {
        switch (status) {
            case ALARM:
                return "alarm";
            case MISUNDERSTANDING:
                return "misunderstanding";
            case SCARED:
                return "scared";
            case NOTPRESENT:
                return "not present";
            case PRESENT:
                return "present";
            case BIZARRE:
                return "bizarre";
            case DISTRAUGHT:
                return "distraught";
            case INDISTINCT:
                return "indistinct";
            case SAD:
                return "sad";
            case LOUD:
                return "loud";
            default:
                return "";
        }
    }
    default String sizeToString(Size size) {
        switch (size) {
            case BIG:
                return "big";
            case SMALL:
                return "small";
            case MIDDLE:
                return "middle";
            default:
                return "";
        }
    }

    default String typeToString(TypeOfMovement type) {
        switch (type) {
            case WALK:
                return "walk";
            case FLY:
                return "fly";
            case RUN:
                return "run";
            case INACTION:
                return "inaction";
            default:
                return "";
        }
    }
```

```
    default String roomToString(Flat type) {
        switch (type) {
            case KID_ROOM:
                return "Kid's room";
            case KITCHEN:
                return "kitchen";
            case BATHROOM:
                return "bathroom";
            case BETAN_ROOM:
                return "Betan's room";
            case BOSSE_ROOM:
                return "Bosse's room";
            case HALL:
                return "hall";
            case DINING_ROOM:
                return "dining room";
            case PARENTS_ROOM:
                return "parents's room";
            default:
                return "";
        }
    }
}
```

Flat.java

```
public enum Flat {
    KITCHEN,
    BATHROOM,
    LIVING_ROOM,
    HALL,
    DINING_ROOM,
    BOSSE_ROOM,
    BETAN_ROOM,
    KID_ROOM,
    PARENTS_ROOM,
}
```

Furniture.java

```
public enum Furniture {
    WINDOW,
    BED,
    BOOKCASE,
    TABLE,
    CHAIR,
    COMMODE,
    DOOR,
    BATH,
    SOFA,
    TAZ_S_VODOY
}
```

Gender.java

```
public enum Gender {
    MALE,
    FEMALE
}
```

Size.java

```
public enum Size {
    SMALL,
    BIG,
    MIDDLE
}
```

Status.java

```java
public enum Status {
    MISUNDERSTANDING,  //непонимание
    SCARED,   //испуганный
    ALARM,  //тревожный
    PRESENT, //настоящий
    NOTPRESENT, //ненастоящий
    NORMAL,//нормальный
    DISTRAUGHT, //обезумевший
    BIZARRE,   //причудливый
    INDISTINCT,    //глухой(звук)
    SAD,   //печальный
    LOUD,  //громко

}
```

TypeOfMovement.java

```java
public enum TypeOfMovement {
    FLY,
    WALK,
    RUN,
    INACTION
}
```

NotFlatRunTimeException.java

```java
public class NotFlatRunTimeException extends RuntimeException{
    public NotFlatRunTimeException() { super("Warning! Wrong Flat!"); }
}
```

NotGenderException.java

```java
public class NotGenderException extends ReflectiveOperationException {
    public NotGenderException(String message) { super(message); }
}
```

NullObjException.java

```java
public class NullObjException extends RuntimeException {
    public NullObjException(String message) { super(message); }
}
```

RoomNotFoundException.java

```java
public class RoomNotFoundException extends ReflectiveOperationException{
    public RoomNotFoundException(String message) { super(message); }
}
```

Результат работы программы:

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Family Svanteson was created.
The window in the bathroom looks out onto NO.
The window in the Betan's room looks out onto NO.
The window in the Bosse's room looks out onto at street.
The window in the parents's room looks out onto at garden.
The window in the hall looks out onto NO.
The window in the kitchen looks out onto at garden.
The window in the Kid's room looks out onto at street.
The window in the dining room looks out onto at street.
Freken Bok change status to alarm.
Malysh tried to calm down Freken Bok
Freken Bok could not calm down.
Freken Bok didn't listen to Malysh
Freken Bok's alarm Sight  were fixed on the .
Freken Bok follow at small ghost from Vasastan
small ghost from Vasastan fly bizarre
It circled in the night, receded, approached again, now soaring up, then descending below, and from time to time did a little somersault in the air.
Freken Bok follow at small ghost from Vasastan
sad sounds did not subside for a moment.
'A little white ghost, a dark starry sky, sad music - how beautiful and interesting all this is!' - thought Malysh
Freken Bok  cling to  Malysh.
Malysh resisted:
'you can't allow everything to break down now, after such a good start!' - thought Malysh
Freken Bok stubbornly stood her ground.
Malysh could not understand in any way. Freken Bok.
Malysh sat on bed.
Freken Bok change status to scared.
Malysh look at scared Freken Bok
Malysh shook his head.
Freken Bok did not want to hear about Frida.
Freken Bok continued to move the all furnitures.
Freken Bok move commode to door.
Freken Bok move table to door.
Freken Bok move chair to door.
Freken Bok move bookcase to door.
A real barricade has been created.
indistinct sounds.
small ghost from Vasastan says:' sounded more satisfying'.
small ghost from Vasastan swiftly, with a whistle flew out from under bed.
small ghost from Vasastan burst into a long, dull laugh.
Freken Bok was not laughing.
Freken Bok rushed to door.
Freken Bok began to throw furniture.
Freken Bok dismantled in the blink of an eye the Barricade.
Freken Bok run from parents's room to hall
Freken Bok with a loud cry, she ran into the hall.
small ghost from Vasastan fly from parents's room to hall
Malysh run from parents's room to hall
Bimbo run from parents's room to hall
Bimbo says:' barked'.
Bimbo recognized by smell small ghost from Vasastan.
'a fun game has begun!' - thought Bimbo
'a fun game has begun!' - thought small ghost from Vasastan
small ghost from Vasastan lagged behind a bit to make a real chase.
They ran all over the apartment.
Freken Bok run from hall to kitchen
small ghost from Vasastan fly from hall to kitchen
Freken Bok run from kitchen to dining room
small ghost from Vasastan fly from kitchen to dining room
Freken Bok run from dining room to Kid's room
small ghost from Vasastan fly from dining room to Kid's room
Freken Bok run from Kid's room to kitchen
small ghost from Vasastan fly from Kid's room to kitchen
Freken Bok run from kitchen to hall
small ghost from Vasastan fly from kitchen to hall
Freken Bok run from hall to Kid's room
small ghost from Vasastan fly from hall to Kid's room
Freken Bok says:' screamed'.
small ghost from Vasastan tried to calm down Freken Bok
Freken Bok could not calm down.
Freken Bok continued to scream and rush around the kitchen.
There still stood on the floora basin of water in which she washed her feet.
small ghost from Vasastan chase Freken Bok
small ghost from Vasastan loud says: 'Gey, gey!'.
Freken Bok tripped over a basin of water in which she washed her feet.
Freken Bok fell with a crash.
Freken Bok loud says: 'a scream like a siren'.

Process finished with exit code 0
```

Вывод:

В ходе выполнения данной лабораторной работы я познакомилась с внутренними классами (вложенные, анонимные, локальные) в языке Java. Узнала, что такое классы-исключения. Научилась создавать собственные классы-исключения.