Национальный исследовательский университет ИТМО

Факультет СУиР

Лабораторная работа № 4

Работу выполнил:
Малышева Анна

Группа № R3137

Преподаватель:
Райла Мартин

г. Санкт-Петербург

2020

Вариант: 373052

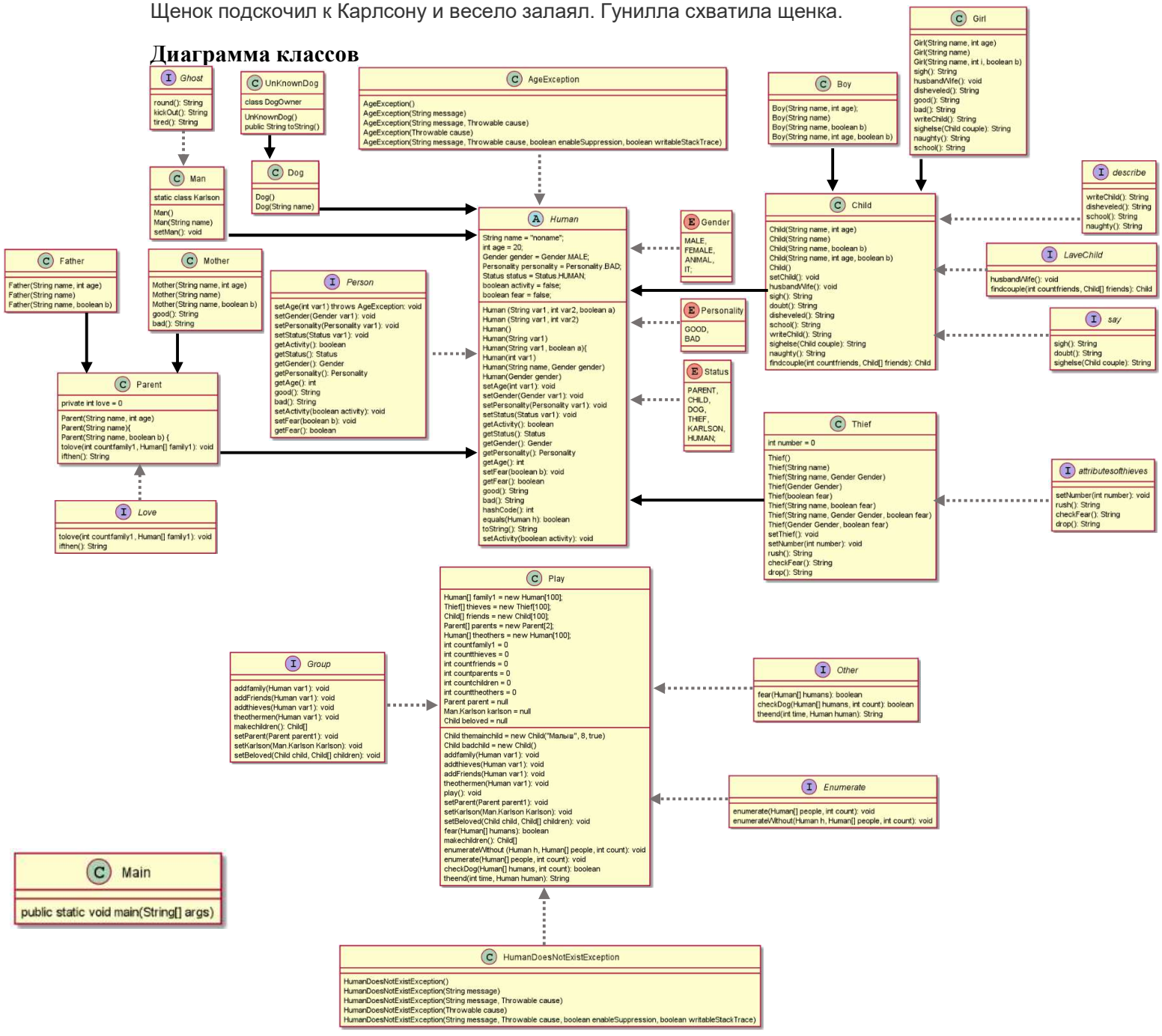
Задание:

Описание предметной области, по которой должна быть построена объектная модель:

Рулле и Филле бросились к двери, а привидение вилось вокруг них. Не помня себя от страха, они выскочили в прихожую, а оттуда на лестничную площадку. Привидение преследовало их по пятам, гнало вниз по лестнице и выкрикивало время от времени глухим, страшным голосом: Но потом привидение устало и вернулось в столовую. Малыш собрал с пола деньги, кольца, брошки и положил все это обратно в секретер. А Гунилла и Кристер подобрали все вилки и ложки, которые уронил Филле, когда он метался между кухней и столовой. Дети смеялись; они были счастливы. А Карлсон добавил: Малыш прыгал от радости, что все обернулось так хорошо. На следующее утро, едва проснувшись, взъерошенный мальчуган в полосатой голубой пижаме пришлепал босиком к маме на кухню. Папа уже ушел на службу, а Боссе и Бетан -- в школу. У Малыша уроки начинались позже, и это было очень кстати, потому что он любил оставаться вот так по утрам вдвоем с мамой, пусть даже ненадолго. В такие минуты хорошо разговаривать, вместе петь песни или рассказывать друг другу сказки. Хотя Малыш уже большой мальчик и ходит в школу, он с удовольствием сидит у мамы на коленях, но только если этого никто не видит. Когда Малыш вошел в кухню, мама, примостившись у кухонного стола, читала газету и пила кофе. Малыш молча влез к ней на колени. Мама обняла его и нежно прижала к себе. Так они и сидели, пока Малыш окончательно не проснулся. Мама и папа вернулись вчера с прогулки позже, чем предполагали. Малыш уже лежал в своей кроватке и спал. Во сне он разметался. Укрывая его, мама заметила дырки, прорезанные в простыне. А сама простыня была такая грязная, словно ее кто-то нарочно исчертил углем. И тогда мама подумала: "Неудивительно, что Малыш поспешил лечь спать". А теперь, когда озорник сидел у нее на коленях, она твердо решила не отпускать его без объяснений. Малыш молчал и напряженно думал. Как быть? Ведь дырки прорезал именно Карлсон, а мама запретила о нем говорить. Малыш решил также ничего не рассказывать и о ворах, потому что мама все равно этому не поверит. "А! Значит, это Гунилла разрезала простыню", -- подумала мама. И еще она подумала, что ее Малыш -- хороший мальчик, потому что он не желает наговаривать на других, а хочет, чтобы Гунилла сама все рассказала. Мама обняла Малыша за плечи. Она решила сейчас больше ни о чем его не расспрашивать, но при случае поговорить с Гуниллой. Мама вновь принялась читать газету, а Малыш молча сидел у нее на коленях и думал. Кого же, собственно говоря, он действительно любит? Прежде всего маму... и папу тоже... Еще он любит Боссе и Бетан... Ну да, чаще всего он их все-таки любит, особенно Боссе. Но иногда он на них так сердится, что вся любовь пропадает. Любит он и Карлсона, который живет на крыше, и Гуниллу тоже любит. Да, быть может, он женится на ней, когда вырастет, потому что хочешь не хочешь, а жену иметь надо. Конечно, больше всего он хотел бы жениться на маме, но ведь это невозможно. Вдруг Малышу пришла в голову мысль, которая его встревожила. Мама подвинула к себе чашку и с удивлением взглянула на Малыша. Малыш, испугавшись, что сморозил глупость, решил не продолжать. Но мама настаивала: Да, это было так. Тут мама рассмеялась и сказала: И Малыш вновь задумался. Он думал о том, что ему, наверно, будет не очень приятно жить вместе с Гуниллой, потому что с ней иногда трудно ладить. Да и вообще ему больше всего хотелось жить вместе с мамой, папой, Боссе и Бетан, а не с какой-то там женой. Мама вздохнула. Ну вот, опять Малыш заговорил о своей вожделенной собаке! Это было почти так же невыносимо, как и разговоры о Карлсоне, который живет на крыше. ...В тот день Малышу было приятно идти в школу, потому что ему многое надо было обсудить с Кристером и Гуниллой. Домой они шли, как всегда, вместе. И Малыша это особенно радовало, потому что теперь Кристер и Гунилла тоже были знакомы с Карлсоном. Тут появилось еще одно существо, которое тоже захотело пойти вместе с ними. Когда ребята собрались перейти улицу, к Малышу подбежал маленький черный пудель. Он обнюхал коленки Малыша и дружески тявкнул. Малыш был бы счастлив переводить щенка через все перекрестки города. Должно быть, щенок это почувствовал: он бежал вприпрыжку по

мостовой, норовя прижаться к ноге Малыша. У маленького щенка был такой вид, будто он готов любить всех на свете, только бы его любили. И Малыш полюбил этого щенка. О, как он его полюбил! Он нагнулся к щенку и принялся его ласкать, и гладить, и тихонько присвистывать, и причмокивать. Все эти нежные звуки должны были означать, что черный пудель -- самый симпатичный, самый распрекрасный пес на свете. Щенок вилял хвостом, всячески давая понять, что он тоже так думает. Он радостно прыгал и лаял, а когда дети свернули на свою улицу, побежал вслед за ними. Разве мог понять Кристер, у которого был Ефа, что значит не иметь собаки -- совсем никакой собаки! И щенок пошел за ним. Так он оказался у дверей дома, где жил Малыш. Тут Малыш взял его на руки и понес по лестнице. "Сейчас я спрошу у мамы, можно ли мне оставить его у себя". Но мамы не было дома. В записке, которую Малыш нашел на кухонном столе, было сказано, что она в прачечной и что он может туда зайти, если ему что-нибудь понадобится. Тем временем щенок, как ракета, ворвался в комнату Малыша. Ребята помчались за ним. В эту самую минуту в окно влетел Карлсон, который живет на крыше. Мальты не слушал Карлсона. Тысячи летающих собак ничего не значили для него по сравнению с этим маленьким милым щенком. Гунилла склонилась над собакой. Но те лишь обидно рассмеялись в ответ. Щенок подскочил к Карлсону и весело залаял. Гунилла схватила щенка.

Диаграмма классов



Исходный код программы

1) children

a) Boy

```
package children;

import human.AgeException;
import human.Gender;

public class Boy extends Child {
    public Boy(String name, int age) throws AgeException {
        super(name, age);
        setGender(Gender.MALE);
    }

    public Boy(String name) {
        super(name);
        setGender(Gender.MALE);
    }
    public Boy(String name, boolean b) {
        super(name, b);
        setGender(Gender.MALE);
    }

    public Boy(String name, int age, boolean b) throws AgeException {
        super(name, age, b);
        setGender(Gender.MALE);
    }

}
```

b) Child

```
package children;

import human.*;

public class Child extends Human implements LoveChild, describe, say {
    // Child
    public Child(String name, int age) throws AgeException {
        super(name, age);
        this.setChild();
    }
    public Child(String name){
        super(name);
        this.setChild();
    }
    public Child(String name, boolean b) {
        super(name, b);
        this.setChild();
    }
    public Child(String name, int age, boolean b) throws AgeException {
        super(name, age, b);
        this.setChild();
    }

    public Child() {
        this.setChild();
    }
```

```
    public void setChild(){
        this.setStatus(Status.CHILD);
        this.setGender(Gender.MALE);
        if (this.getAge() < 13) {
            this.setPersonality(Personality.GOOD);
        }
    }
    public void husbandWife() {
        System.out.print(", а не с какой-то там женой. ");
    }
    public String sigh() {
        return "Ну, тогда я ни на ком не женюсь, - вздохнул " + this.toString()
+ ".\n";
    }
    public String doubt(){
        return ", наверное,";
    }
    public String disheveled(){
        return "взъерошенный мальчуган";
    }
    public String school(){
        if (getAge() < 7){
            return this.toString() + " еще ходил в детский садик, и это было
очень кстати, потому что ";
        }else{
            if (getAge() < 13){
                return "У " + this.toString() + " уроки начинались позже, и это
было очень кстати, потому что ";
            }
            else{
                return this.toString() + " обычно вставал раньше, чем надо было,
потому что ";
            }
        }
    }
    public String writeChild(){
        return "мальчик";
    }
    public String sighelse(Child couple){
        return "- Ну, тогда я женюсь на " + couple.toString() + ", - вздохнул "
+ this.toString() + ". - Ведь надо же мне будет на ком-нибудь жениться!\n";
    }
    public String naughty(){
        return "озорник";
    }
    public Child findcouple(int countfriends, Child[] friends){
        Child abstract_human = null;
        for (int i = 0; i < countfriends; i++){
            if (friends[i].getGender() != this.getGender()){
                abstract_human = friends[i];
                break;
            }
        }
        return abstract_human;
    }

}
```

c) describe

```
package children;

public interface describe {
    String writeChild();
    String disheveled();
    String school();
    String naughty();
}
```

d) Girl

```
package children;

import human.AgeException;
import human.Gender;

public class Girl extends Child {
    public Girl(String name, int age) throws AgeException {
        super(name, age);
        setGender(Gender.FEMALE);
    }

    public Girl(String name) {
        super(name);
        setGender(Gender.FEMALE);
    }

    public Girl(String name, int i, boolean b) throws AgeException {
        super(name, i, b);
        this.setGender(Gender.FEMALE);
    }

    @Override
    public String sigh() {
        return "Ну, тогда я ни за кого не выйду замуж, - вздохнула " +
this.toString() + ".\n";
    }

    @Override
    public void husbandWife(){
        System.out.print(", а не с каким-то там мужем. ");
    }

    @Override
    public String disheveled(){
        return "взъерошенная девчушка";
    }

    @Override
    public String good(){
        return "хорошая";
    }

    @Override
    public String bad(){
        return "плохая";
    }
    @Override
    public String writeChild(){
        return "девочка";
```

```
    }
    @Override
    public String sighelse(Child couple) {
        return "- Ну, тогда я выйду замуж за " + couple.toString() + ", -
вздохнула " + this.toString() + ". - Ведь надо же мне будет выйти замуж за кого-
нибудь!\n";
    }
    @Override
    public String naughty(){
        return "озорница";
    }
    @Override
    public String school(){
        if (getAge() < 7){
            return this.toString() + " еще ходила в детский садик, и это было
очень кстати, потому что ";
        }else{
            if (getAge() < 13){
                return "У " + this.toString() + " уроки начинались позже, и это
было очень кстати, потому что ";
            }
            else{
                return this.toString() + "обычно вставала раньше, чем надо было,
потому что ";
            }
        }
    }
}
```

e) LoveChild

```
package children;

public interface LoveChild {
    void husbandWife();
    Child findcouple(int countfriends, Child[] friends);
}
```

f) say

```
package children;

public interface say {
    String sigh();
    String doubt();
    String sighelse(Child couple);
}
```

2) dog

a) Dog

```
package dog;

import human.Gender;
import human.Human;
import human.Status;

public class Dog extends Human{

    public Dog(){
        this.setGender(Gender.ANIMAL);
        this.setStatus(Status.DOG);
```

```
    }
    public Dog(String name){
        super(name);
        this.setGender(Gender.ANIMAL);
        this.setStatus(Status.DOG);
    }

}
```

b) UnKnownDog

```
package dog;

public class UnKnownDog {
    public UnKnownDog(){
    }
    @Override
    public String toString(){
        return "черный пудель";
    }
    public class DogOwner{
        String name = "владелица собаки";
        public DogOwner(){
        }
        public DogOwner(String name){
            this.name = name;
        }
        @Override
        public String toString(){
            return "DOGOWNER " + name;
        }
    }
}
```

3) human

a) AgeException

```
package human;

public class AgeException extends RuntimeException{
    public AgeException() {
    }

    public AgeException(String message) {
        super(message);
    }

    public AgeException(String message, Throwable cause) {
        super(message, cause);
    }

    public AgeException(Throwable cause) {
        super(cause);
    }

    public AgeException(String message, Throwable cause, boolean
enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }
}
```

b) Gender

```
package human;

public enum Gender {
    MALE,
    FEMALE,
    ANIMAL,
    IT;
}
```

c) Human

```
package human;

public abstract class Human implements Person {

    private String name = "noname";
    private int age = 20;
    private Gender gender = Gender.MALE;
    private Personality personality = Personality.BAD;
    private Status status = Status.HUMAN;
    private boolean activity = false;
    private boolean fear = false;

    // Human
    public Human (String var1, int var2, boolean a) {
        this.name = var1;
        try {
            setAge(var2);
        }catch(AgeException e){
            System.err.println(e.getMessage());
        }
        this.activity = a;
    }

    public Human (String var1, int var2) throws AgeException {
        this.name = var1;
        try {
            setAge(var2);
        }catch(AgeException e){
            System.err.println(e.getMessage());
        }
    }

    public Human(){
    }

    public Human(String var1){
        this.name = var1;
    }

    public Human(String var1, boolean a){
        this.name = var1;
        this.activity = a;
    }

    public Human(int var1){
    }

    public Human(String name, Gender gender) {
        this.setGender(gender);
```

```
        this.name = name;
    }

    public Human(Gender gender) {
        this.setGender(gender);
    }

    // из интерфейса Person
    public void setAge(int var1) throws AgeException {
        if ((var1 < 0) || (var1 > 99)){
            throw new AgeException("Age of " + this.name + " is incorrect: " +
var1 + ". The corrected age: " + this.age);
        }
        this.age = var1;
    }

    public void setGender(Gender var1){
        if (var1 == null){
            var1 = Gender.MALE;
        }
        this.gender = var1;
    }

    public void setPersonality(Personality var1){
        if (var1 == null){
            var1 = Personality.BAD;
        }
        this.personality = var1;
    }

    public void setStatus(Status var1){
        if (var1 == null){
            var1 = Status.CHILD;
        }
        this.status = var1;
    }

    public boolean getActivity(){
        return(this.activity);
    }

    public Status getStatus(){ return this.status; }

    public Gender getGender(){ return this.gender;}

    public Personality getPersonality(){
        return this.personality;
    }

    public int getAge(){ return this.age; }

    public void setFear(boolean b){
        this.fear = b;
    }

    public boolean getFear(){
        return this.fear;
    }
```

```
    public String good(){
        return "хороший";
    }

    public String bad(){
        return "плохой";
    }

    @Override
    public int hashCode(){
        int a = this.age;
        if (this.gender == Gender.FEMALE){
            a += 100;
        }
        if (this.gender == Gender.MALE){
            a += 200;
        }
        if (this.status == Status.CHILD){
            a += 1000;
        }
        if (this.status == Status.PARENT){
            a += 2000;
        }
        return a;
    }

    public boolean equals(Human h){
        return (this.hashCode() != h.hashCode());
    }

    @Override
    public String toString(){
        String stringstatus = String.valueOf(this.status);
        return (stringstatus + " " + this.name);
    }

    public void setActivity(boolean activity){
        this.activity = activity;
    }
}
```

d) Person

```
package human;

public interface Person {
    void setAge(int var1) throws AgeException;
    void setGender(Gender var1);
    void setPersonality(Personality var1);
    void setStatus(Status var1);
    boolean getActivity();
    Status getStatus();
    Gender getGender();
    Personality getPersonality();
    int getAge();
    String good();
    String bad();
    void setActivity(boolean activity);
    void setFear(boolean b);
```

```
    boolean getFear();
}
```

e) Personality

```
package human;

public enum Personality {
    GOOD,
    BAD
}
```

f) Status

```
package human;

public enum Status {
    PARENT,
    CHILD,
    DOG,
    THIEF,
    KARLSON,
    HUMAN;
}
```

4) karlson

a) Ghost

```
package karlson;

public interface Ghost {
    default String round(){
        return "а приведение вилось вокруг ";
    }
    default String kickOut(){
        return "Привидение преследовало по пятам, гнало вниз по лестнице и
выкрикивало время от времени глухим, страшным голосом: ";
    }
    default String tired(){
        return "Но потом привидение устало и вернулось в столовую. ";
    }
}
```

b) Man

```
package karlson;

import human.*;

public class Man extends Human implements Ghost {
    public Man() throws AgeException {
        this.setMan();
    }
    public Man(String name) throws AgeException {
        super(name);
        this.setMan();
    }
    public void setMan() throws AgeException {
        this.setAge(30);
        if (this.getActivity()) {
            this.setPersonality(Personality.GOOD);
        }
        this.setGender(Gender.MALE);
```

```
    }
    public static class Karlson extends Man{
        public Karlson(String name) throws AgeException {
            super(name);
            this.setAge(30);
            if (this.getActivity()) {
                this.setPersonality(Personality.GOOD);
            }
            this.setGender(Gender.MALE);
            this.setStatus(Status.KARLSON);
        }
    }
}
```

5) parents

a) Father

```
package parents;

import human.AgeException;
import human.Gender;

public class Father extends Parent {
    public Father(String name, int age) throws AgeException {
        super(name, age);
        setGender(Gender.MALE);
    }

    public Father(String name) {
        super(name);
        setGender(Gender.MALE);
    }

    public Father(String name, boolean b) {
        super(name, b);
        setGender(Gender.MALE);
    }
}
```

b) Love

```
package parents;

import human.Human;

public interface Love {
    void tolove(int countfamily1, Human[] family1);
    String ifthen();
}
```

c) Mother

```
package parents;

import human.AgeException;
import human.Gender;

public class Mother extends Parent {

    public Mother(String name, int age) throws AgeException {
        super(name, age);
        setGender(Gender.FEMALE);
```

```
    }

    public Mother(String name) {
        super(name);
        setGender(Gender.FEMALE);
    }

    public Mother(String name, boolean b) {
        super(name, b);
        setGender(Gender.FEMALE);
    }

    @Override
    public String good(){
        return "хорошая.";
    }

    @Override
    public String bad(){
        return "плохая.";
    }

}
```

d) Parent

```
package parents;

import human.AgeException;
import human.Human;
import human.Personality;
import human.Status;

public class Parent extends Human implements Love {
    private int love = 0;

    //Parent
    public Parent(String name, int age) throws AgeException {
        super(name, age);
        this.setStatus(Status.PARENT);
    }
    public Parent(String name){
        super(name);
        this.setStatus(Status.PARENT);
    }

    public Parent(String name, boolean b) {
        super(name, b);
        this.setStatus(Status.PARENT);
        this.setPersonality(Personality.GOOD);
    }
    public void tolove(int countfamily1, Human[] family1){
        for (int j = 0; j < countfamily1; j++) {
            if (((family1[j].hashCode() > 2000) && (family1[j].getGender() !=
this.getGender())) || ((family1[j].getStatus() == Status.CHILD) &&
((family1[j].getPersonality() == Personality.GOOD) ||
(family1[j].getActivity())))) {
                love += 1;
            }
        }
```

```
    }
    public String ifthen() {
        String s = "";
        if (this.getActivity()) {
            switch (love) {
                case 1:
                    s = "- Раз ты меня любишь, значит я ";
                    s += this.good();
                    break;
                case 2:
                    s = "- Раз вы оба меня любите, значит я ";
                    s += this.good();
                    break;
                case 0:
                    s = "- Раз никто меня не любит, значит я ";
                    s += this.bad();
                    break;
                default:
                    s = "- Раз много людей меня любят, значит я ";
                    s += this.good();
                    break;
            }
        }
        return s;
    }
}
```

6) themain

a) Enumerate

```
package themain;

import human.Human;

public interface Enumerate {
    void enumerate(Human[] people, int count);
    void enumerateWithout(Human h, Human[] people, int count);
}
```

b) Group

```
package themain;

import children.Child;
import human.Human;
import karlson.Man;
import parents.Parent;

public interface Group {
    void addfamily(Human var1);
    void addFriends(Human var1);
    void addthieves(Human var1);
    void theothermen(Human var1);
    Child[] makechildren();
    void setParent(Parent parent1) throws HumanDoesNotExistException;
    void setKarlson(Man.Karlson Karlson) throws HumanDoesNotExistException;
    void setBeloved(Child child, Child[] children) throws
HumanDoesNotExistException;
}
```

c) HumanDoesNotExistException

```
package themain;

import karlson.Man;

public class HumanDoesNotExistException extends Exception{
    public HumanDoesNotExistException() {
    }

    public HumanDoesNotExistException(String message) {
        super(message);
    }

    public HumanDoesNotExistException(String message, Throwable cause) {
        super(message, cause);
    }

    public HumanDoesNotExistException(Throwable cause) {
        super(cause);
    }

    public HumanDoesNotExistException(String message, Throwable cause, boolean
enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }
}
```

d) Main

```
package themain;

import children.Boy;
import children.Girl;
import dog.Dog;
import human.AgeException;
import karlson.Man;
import parents.Father;
import parents.Mother;
import thieves.Thief;

public class Main {

    public static void main(String[] args) throws AgeException,
HumanDoesNotExistException {
        Play p = new Play();
        Thief thief1 = new Thief("Рулле");
        Thief thief2 = new Thief("Филле");
        Mother mother = new Mother("Мама", true);
        Father father = new Father("Папа");
        Boy junior = new Boy("Малыш", 20, true);
        Girl Gunilla = new Girl("Гунилла");
        Boy Krister = new Boy("Кристер");
        Dog dog = new Dog("Бимбо");
        Boy Bosse = new Boy("Боссе");
        Girl Betan = new Girl("Бетан");
        Man.Karlson Karlson = new Man.Karlson("Карлсон, который живет на
крыше");

        p.addthieves(thief1);
        p.addthieves(thief2);
        p.addfamily(mother);
```

```
        p.addfamily(father);
        p.addfamily(junior);
        p.addfamily(Bosse);
        p.addfamily(Betan);
        p.addfamily(dog);
        p.addFriends(Gunilla);
        p.addFriends(Krister);
        p.theothermen(Karlson);

        p.play();
    }
}
```

e) Other

```
package themain;

import human.Human;

public interface Other {
    boolean fear(Human[] humans);
    boolean checkDog(Human[] humans, int count);
    String theend(int time, Human human);
}
```

f) Play

```
package themain;

import children.Child;
import dog.UnKnownDog;
import human.*;
import parents.Mother;
import thieves.Thief;
import karlson.Man;
import parents.Parent;

public class Play implements Group, Enumerate, Other {

    private final Human[] family1 = new Human[100];
    private final Thief[] thieves = new Thief[100];
    private final Child[] friends = new Child[100];
    private final Parent[] parents = new Parent[2];
    private final Human[] theothers = new Human[100];

    private int countfamily1 = 0;
    private int countthieves = 0;
    private int countfriends = 0;
    private int countparents = 0;
    private int countchildren = 0;
    private int counttheothers = 0;

    private Child themainchild = new Child("Малыш", 8, true);
    private Child badchild = new Child();
    private Parent parent = null;
    private Man.Karlson karlson = null;
    private Child beloved = null;

    public void addfamily(Human var1) {
        family1[countfamily1] = var1;
        if (family1[countfamily1].hashCode() > 2000){
```

```
            if (parents[0] == null) {
                parents[0] = (Parent) family1[countfamily1];
                countparents = 1;
            }else{
                if (family1[countfamily1].getGender() !=
parents[0].getGender()){
                    parents[1] = (Parent) family1[countfamily1];
                    countparents = 2;
                }
            }
        }
        countfamily1++;
    }

    public void addthieves(Human var1) {
        thieves[countthieves] = (Thief) var1;
        countthieves++;
    }

    public void addFriends(Human var1) {
        friends[countfriends] = (Child) var1;
        countfriends++;
    }
    public void theothermen(Human var1){
        theothers[counttheothers++] = var1;
    }

    public void play() {
        Child[] children = makechildren();
        // Главный герой
        for (int i = 0; i < countchildren; i++) {
            if (children[i].getActivity()) {
                themainchild = children[i];
                break;
            }
        }
        // Гунилла (первый плохой ребенок из массива детей)
        for (int i = 0; i < countchildren; i++) {
            if (friends[i].getPersonality() == Personality.BAD) {
                badchild = friends[i];
                break;
            }
        }
        // Главный родитель героя рассказа (по умолчанию - мама)
        Parent parent1 = null;
        for (int i = 0; i < countparents; i++) {
            if (parents[i].getActivity()) {
                parent1 = parents[i];
                break;
            }
        }
        try {
            setParent(parent1);
        }catch (HumanDoesNotExistException e){
            System.err.println(e.getMessage());
        }
        // Братья и сестры (только школьники)
        Child[] schoolchildren = new Child[100];
        int countsiblings = 0;
```

```
        for (int j = 0; j < countfamily1; j++) {
            if ((family1[j].getStatus() == Status.CHILD) && (family1[j].getAge()
> 6) && ((themainchild.equals(family1[j])) || (!
themainchild.toString().equals(family1[j].toString())))) {
                schoolchildren[countsiblings++] = (Child) family1[j];
            }
        }
        class GroupOfThieves extends Thief {
            public GroupOfThieves(boolean fearthieves, int countthieves){
                setFear(fearthieves);
                setNumber(countthieves);
                if (countthieves == 1){
                    setGender(thieves[0].getGender());
                }
            }
        }
        Man.Karlson Karlson = null;
        for (int i = 0; i < counttheothers; i++){
            if (theothers[i].getStatus() == Status.KARLSON){
                Karlson = (Man.Karlson)theothers[i];
            }
        }
        try {
            setKarlson(Karlson);
        }catch(HumanDoesNotExistException e){
            parents[0] = parent;
            System.err.println(e.getMessage());
        }
        Child Beloved = null;
        for (int i = 0; i < countfamily1; i++) {
            if ((family1[i].hashCode() < 2000) &&
(family1[i].equals(themainchild))) {
                if (family1[i].getGender() == Gender.FEMALE) {
                    Beloved = (Child)family1[i];
                }
            }
        }
        try {
            setBeloved(Beloved, children);
        }catch(HumanDoesNotExistException e){
            System.err.println(e.getMessage());
        }
        // Сцена с жуликами
        if (countthieves > 0) {
            GroupOfThieves groupofthieves = new GroupOfThieves(fear(thieves),
countthieves);
            enumerate(thieves, countthieves);
            System.out.print(" " + groupofthieves.rush() + karlson.round() + "н"
+ theend(2, groupofthieves) + ". " + groupofthieves.checkFear());
            if (fear(thieves)) {
                System.out.print(karlson.kickOut() + karlson.tired() +
themainchild.toString() + " собрал" + theend(1, themainchild) + " с пола деньги,
кольца, брошки и положил" + theend(1, themainchild) + " все это обратно в
секретер. ");
                enumerate(friends, countfriends);
                String s = "и";
                if (countfriends == 0){
                    System.out.print(themainchild.toString());
                    s = theend(1, themainchild);
```

```
                }
                if (countfriends == 1){
                    s = theend(1, friends[0]);
                }
                System.out.print(" подобрал" + s + " все вилки и ложки, " +
groupofthieves.drop() + "Дети смеялись; они были счастливы. " +
themainchild.toString() + " прыгал" + theend(1, themainchild) + " от радости,
что все обернулось так хорошо. ");
            } else {
                System.out.print(karlson.tired() + "Все были расстроены
произошедшим. ");
            }
        }else{
            System.out.print(karlson.toString() + " решил притвориться
приведением. Для этого он взял простыню, нарисовал на ней \"лицо\" приведения и
пошел пугать детей. Все смеялись. ");
        }
        System.out.println();
        // На следующее утро... до разговора
        System.out.print("На следующее утро, едва проснувшись, " +
themainchild.disheveled() + " в полосатой голубой пижаме пришлепал" + theend(1,
themainchild) + " босиком к " + parent.toString() + " на кухню. ");
        // Все ушли, кроме главного героя и parent
        if (countparents > 1) {
            System.out.print(parents[1].toString() + " уже уш" + theend(4,
parents[1]) + " на службу. ");
        }
        enumerate(schoolchildren, countsiblings);
        if (countsiblings > 0) {
            System.out.print(" - в школу. ");
        }
        System.out.print(themainchild.school() + "он" + theend(1, themainchild)
+ " любил" + theend(1, themainchild) + " оставаться вот так по утрам вдвоем с "
+ parent.toString() + ", пусть даже ненадолго. В такие минуты хорошо
разговаривать, вместе петь песни или рассказывать друг другу сказки. ");
        if (themainchild.getAge() < 7){
            System.out.print("Он" + theend(1, themainchild) + " еще маленьк" +
theend(10, themainchild) + ", и поэтому с удовольствием сидит у " +
parent.toString() + " на коленках.");
        }else{
            if (themainchild.getAge() < 13){
                System.out.print("Хотя " + themainchild.toString() + " уже
больш" + theend(11, themainchild) + " " + themainchild.writeChild() + " и ходит
в школу, он" + theend(1, themainchild) + " с удовольствием сидит у " +
parent.toString() + " на коленях, но только если этого никто не видит. ");
            }
        }
        System.out.println();
        // Малыш вышел на кухню
        System.out.print("Когда " + themainchild.toString() + " вош" + theend(4,
themainchild) + " в кухню, " + parent.toString() + ", примостившись у кухонного
стола, пил" + theend(1, parent) + " кофе и читал" + theend(1, parent) + "
газету. ");
        if (themainchild.getAge() < 13){
            System.out.print(themainchild.toString() + " молча влез" + theend(6,
themainchild) + " на колени. " + parent.toString() + " обнял" + theend(1,
parent) + " " + theend(2, themainchild) + " и нежно прижал" + theend(1, parent)
+ " к себе. Так они и сидели, пока " + themainchild.toString() + " окончательно
не проснул" + theend(3, themainchild) + ". ");
```

```
        }else{
            System.out.print(themainchild.toString() + " сел" + theend(1,
themainchild) + " рядом с " + parent.toString() + ", так они сидели, пока " +
themainchild.toString() + " окончательно не проснул" + theend(3, themainchild) +
". ");
        }
        System.out.println();
        // Обнаружение последствий произошедшего
        enumerate(parents, countparents);
        if (countparents == 2) {
            System.out.print(" вернулись вчера с прогулки позже, чем
предполагали. ");
        }else{
            System.out.print(" вернул" + theend(3, parent) + " вчера с прогулки
позже, чем предполагал" + theend(1, parent) + ". ");
        }
        if (themainchild.getAge() < 13){
            System.out.print(themainchild.toString() + " уже лежал" + theend(1,
themainchild) + " в своей кроватке и спал" + theend(1, themainchild) + ". Во сне
он" + theend(1, themainchild) + " разметал" + theend(3, themainchild) + ".
Укрывая " + theend(2, themainchild) + ", " + parent.toString() + " заметил" +
theend(1, parent) + " дырки, прорезанные в простыне. А сама простыня была такая
грязная, словно ее кто-то нарочно исчертил углем. И тогда " + parent.toString()
+ " подумал" + theend(1, parent) + ": \"Неудивительно, что " +
themainchild.toString() + " поспешил" + theend(1, themainchild)+ " лечь спать.
");
        }else{
            System.out.print(themainchild.toString() + " еще не спал" +
theend(1, themainchild) + ", но было уже очень поздно. Однако " +
parent.toString() + " заметил" + theend(1, parent) + " дырки, прорезанные в
простыне. А сама простыня была такая грязная, словно ее кто-то нарочно исчертил
углем. ");
        }
        // Начало разговора
        if (themainchild.getAge() > 12) {
            System.out.print("А теперь, когда " + themainchild.naughty() + "
приш" + theend(4, themainchild) + " на кухню, он" + theend(1, parent) + " твердо
решил" + theend(1, parent) + " не отпускать " + theend(2, themainchild) + " без
объяснений.");
        } else {
            System.out.print("А теперь, когда " + themainchild.naughty() + "
сидел" + theend(1, themainchild) + " у н" + theend(2, parent) + " на коленях,
он" + theend(1, parent) + " твердо решил" + theend(1, parent) + " не отпускать "
+ theend(2, themainchild) + " без объяснений.");
        }
        System.out.println();
        // Как ответить?
        System.out.print(themainchild.toString() + " молчал" + theend(1,
themainchild) + " и напряженно думал" + theend(1, themainchild) + ". Как быть?
");
        System.out.print("Ведь дырки прорезал именно " + karlson.toString() + ",
а " + parent.toString() + " запретил" + theend(1, parent) + " о нем говорить.
");
        if (countthieves > 0){
            String sthieves = "ах";
            if (countthieves == 1) {
                sthieves = "е";
            }
            System.out.print(themainchild.toString() + " решил" + theend(1,
```

```
themainchild) + " также ничего не рассказывать и о вор" + sthieves + ", потому
что " + parent.toString() + " все равно этому не поверит. ");
        }
        System.out.println();
        // Мама подумала, что это была Гунилла...
        if (!badchild.toString().equals("CHILD noname")) {
            System.out.print("\"А! Значит, это " + badchild.toString() + "
разрезал" + theend(1, badchild) + " простыню\", - подумал" + theend(1, parent) +
" " + parent.toString() + ". И еще он" + theend(1, parent) + " подумал" +
theend(1, parent) + ", что " + theend(2, themainchild) + " " +
themainchild.toString() + " - " + themainchild.good() + " " +
themainchild.writeChild() + ", потому что он" + theend(1, themainchild) + " не
желает наговаривать на других, а хочет, чтобы " + badchild.toString() + " сам" +
theend(1, badchild) + " все рассказал" + theend(1, badchild) + ". " +
parent.toString() + " обнял" + theend(1, parent) + " " + themainchild.toString()
+ " за плечи. Он" + theend(1, parent) + " решил" + theend(1, parent) + " сейчас
больше ни о чем " + theend(2, themainchild) + " не расспрашивать, но при случае
поговорить с " + badchild.toString() + ". " + parent.toString() + " вновь
принял" + theend(3, parent) + " читать газету, ");
        } else {
            System.out.print("\"А! Не признается! Ничего... Потом как-нибудь
спрошу еще раз, - подумал" + theend(1, parent) + " " + parent.toString() + ",
");
        }
        String where = "у н" + theend(2, parent) + " на коленях";
        if (themainchild.getAge() < 13) {
            where = "на кухне";
        }
        //а Малыш молча сидел у нее на коленях и думал
        System.out.print(" а " + themainchild.toString() + " молча сидел" +
theend(1, themainchild) + " " + where + " и думал" + theend(1, themainchild) +
". ");
        // Кого же любит Малыш?
        System.out.print("Кого же, собственно говоря, он" + theend(1,
themainchild) + " действительно любит? ");
        System.out.print("Прежде всего он" + theend(1, themainchild) + " любит "
+ parent.toString() + "... ");
        if (countparents == 2) {
            System.out.print(" и " + parents[1] + " тоже... ");
        }
        if (countchildren > 1) {
            System.out.print("Еще он" + theend(1, themainchild) + " любит ");
            enumerateWithout(themainchild, children, countchildren);
            System.out.print("... Ну да, чаще всего он их все-таки любит,
особенно " + beloved.toString() + ". Но иногда он" + theend(1, themainchild) + "
на н" + theend(2, beloved) + " так сердится, что вся любовь пропадает. ");
        }
        System.out.print(" Любит он" + theend(1, themainchild) + " и " +
karlson.toString());
        if (!badchild.toString().equals("CHILD noname")) {
            String tomarry = "женится на ней";
            String tohavewifeorhusband = "жену иметь надо";
            String wanttomarry = "жениться на маме";
            if (themainchild.getGender() == Gender.FEMALE) {
                tomarry = "выйдет замуж за него";
                tohavewifeorhusband = "мужа иметь надо";
                wanttomarry = "выйти замуж за папу";
            }
            System.out.print(", и " + badchild.toString() + " тоже любит. Да,
```

```
быть может, он" + theend(1, themainchild) + " " + tomarry + ", когда вырастет,
потому что хочешь не хочешь, а " + tohavewifeorhusband + ". Конечно, больше
всего он" + theend(1, themainchild) + " хотел" + theend(1, themainchild) + " бы
" + wanttomarry + ", но ведь это невозможно. Вдруг " + themainchild.toString() +
" пришла в голову мысль, которая его встревожила. " + parent.toString() + "
подвинул" + theend(1, parent) + " к себе чашку и с удивлением взглянул" +
theend(1, parent) + " на " + themainchild.toString() + ". " +
themainchild.toString() + ", испугавшись, что сморозил" + theend(1,
themainchild) + " глупость, решил" + theend(1, themainchild) + " не продолжать.
Но " + parent.toString() + " настаивал" + theend(1, parent) + ": Да, это было
так. ");
        }
        // Тут мама рассмеялась и сказала: Lab_3
        System.out.println("Тут " + parent.toString()+ " расмеял" + theend(3,
parent) + " и сказал" + theend(1, parent) + ":");
        parent.tolove(countfamily1, family1);
        System.out.println(parent.ifthen());
        Child couple = themainchild.findcouple(countfriends, friends);
        if (couple == null) {
            System.out.print(themainchild.sigh() + "Да и вообще " + theend(7,
themainchild) + "больше всего хотелось жить вмсете с ");
        } else {
            String s = "";
            if (couple.getPersonality() == Personality.BAD) {
                s = ", потому что с н" + theend(5, couple) + " трудно ладить. ";
            } else {
                s += ", хотя с н" + theend(5, couple) + " и дружу. ";
            }
            System.out.print(themainchild.sighelse(couple) +
themainchild.toString() + " задумал" + theend(3, themainchild) + ". Он" +
theend(1, themainchild) + " думал" + theend(1, themainchild) + " о том, что " +
theend(7, themainchild) + themainchild.doubt() + " будет не очень приятно жить
вместе с " + couple.toString() + s + "Да и вообще " + theend(7, themainchild) +
" больше всего хотелось жить вмсете с ");
        }
        enumerateWithout(themainchild, family1, countfamily1);
        themainchild.husbandWife();
        // Мама вздохнула.
        if (checkDog(family1, countfamily1)) {
            System.out.print(parent.toString() + " вздохнул" + theend(1, parent)
+ ". Ну вот, опять " + themainchild.toString() + " заговорил" + theend(1,
themainchild) + " о своей вожделенной собаке! Это было почти так же невыносимо,
как и разговоры о " + karlson.toString() + ". ");
        }
        System.out.println();
        // Малыш с друзьями
        if (countfriends > 0) {
            System.out.print("В тот день " + themainchild.toString() + " было
приятно идти в школу, потому что " + theend(7, themainchild) + " многое надо
было обсудить с ");
            enumerate(friends, countfriends);
            System.out.print(". Домой они шли, как всегда, вместе. И " +
themainchild.toString() + " это особенно радовало, потому что теперь ");
            enumerate(friends, countfriends);
            String s = "и";
            String s1 = "ы";
            if (countfriends == 1){
                s = theend(1, friends[0]);
                s1 = s;
```

```
            }
            System.out.println(" тоже был" + s + " знаком" + s1 + " с " +
karlson + ". ");
        }else{
            System.out.println("Домой " + themainchild.toString() + "ш" +
theend(4, themainchild) + "один. ");
        }
        // Появление собаки
        UnKnownDog Dog = new UnKnownDog();
        UnKnownDog.DogOwner dogOwner = Dog.new DogOwner();
        String himherthem = "ними";
        if (countfriends == 0) {
            himherthem = "ним";
            if (themainchild.getGender() == Gender.FEMALE){
                himherthem = "ней";
            }
        }
        System.out.print("Тут появилось еще одно существо, которое тоже захотело
пойти вместе с " + himherthem + ". Когда ребята собрались перейти улицу, к " +
themainchild.toString() + " подбежал маленький " + Dog.toString() + ". Он
обнюхал коленки " + themainchild.toString() + " и дружески тявкнул. ");
        if (checkDog(family1, countfamily1)){
            System.out.print(themainchild.toString() + " был" + theend(1,
themainchild) + " бы счастлив" + theend(1, themainchild) + " переводить щенка
через все перекрестки города. Должно быть, щенок это почувствовал: он бежал
вприпрыжку по мостовой, норовя прижаться к ноге " + themainchild.toString() + ".
У маленького щенка был такой вид, будто он готов любить всех на свете, только бы
его любили. И " + themainchild.toString() + " полюбил" + theend(1, themainchild)
+ " этого щенка. О, как он" + theend(1, themainchild) + " его полюбил" +
theend(1, themainchild) + "! Он нагнул" + theend(3, themainchild) + " к щенку и
принял" + theend(3, themainchild) + " его ласкать, и гладить, и тихонько
присвистывать, и причмокивать. Все эти нежные звуки должны были означать, что "
+ Dog.toString() + " - самый симпатичный, самый распрекрасный пес на свете.
Щенок вилял хвостом, всячески давая понять, что он тоже так думает. Он радостно
прыгал и лаял, а когда дети свернули на свою улицу, побежал вслед за ними. ");
            if (countfriends > 0){
                String genderfriend2 = "го";
                if (friends[0].getGender() == Gender.FEMALE){
                    genderfriend2 = "й";
                }
                System.out.print("Разве мог" + theend(6, friends[0]) + " понять
" + friends[0].toString() + ", у которо" + genderfriend2 + " был Еффа, что
значит не иметь собаки - совсем никакой собаки! И щенок пошел за н" + theend(5,
themainchild) + ". ");
            }
            // Малыш пришел домой
            System.out.print("Так он оказался у дверей дома, где жил " +
themainchild.toString() + ". Тут " + themainchild.toString() + " взял" +
theend(1, themainchild) + " его на руки и понес" + theend(6, themainchild) + "
по лестнице. \"Сейчас я спрошу у " + parent.toString() + ", можно ли мне
оставить его у себя\". ");
        }else{
            System.out.println("Однако собака была такой большой, что ее
появление не обрадовало " + themainchild.toString() + ", а даже несколько
испугала. К счастью, рядом была ее " + dogOwner.toString() + ". ");
            // Малыш пришел домой
            if (themainchild.getGender() == Gender.MALE){
                System.out.print("Вскоре " + themainchild.toString() + " пришел
домой. ");
```

```
            }else {
                System.out.print("Вскоре " + themainchild.toString() + " пришла
домой. ");
            }
        }
        // Записка от мамы
        System.out.print("Но " + parent.toString() + " не было дома. В записке,
которую " + themainchild.toString() + " наш" + theend(4, themainchild) + " на
кухонном столе, было сказано, что он" + theend(1, parent) + " в прачечной и что
он" + theend(1, themainchild) + " может туда зайти, если " + theend(7,
themainchild) + " что-нибудь понадобится. ");
        // Щенок в комнате Малыша
        if (checkDog(family1, countfamily1)){
            System.out.print("Тем временем щенок, как ракета, ворвался в комнату
" + themainchild.toString() + ". ");
            if (countfriends > 0){
                System.out.print("Ребята помчались за ним. ");
            }else{
                System.out.print("Он" + theend(1, themainchild) + " помчалась за
ним. ");
            }
        }else{
            System.out.print(themainchild.toString() + " вош" + theend(4,
themainchild) + " в свою комнату. ");
        }
        System.out.print("В эту самую минуту в окно влетел " +
karlson.toString() + ". ");
        if (checkDog(family1, countfamily1)){
            System.out.print(themainchild.toString() + " не слушал" + theend(1,
themainchild) + " " + karlson.toString() + ". Тысячи летающих собак ничего не
значили для н" + theend(2, themainchild) + " по сравнению с этим маленьким милым
щенком. " + badchild.toString() + " склонил" + theend(3, badchild) + " над
собакой. Но те лишь обидно рассмеялись в ответ. Щенок подскочил к " +
karlson.toString() + " и весело залаял. " + badchild.toString() + " схватил" +
theend(1, badchild) + " щенка.");
        }
    }
    public void setParent(Parent parent1) throws HumanDoesNotExistException {
        if (parent1 == null){
            if (parents[0] == null) {
                parent = new Mother("Мама", true);
                family1[0] = parent;
                throw new HumanDoesNotExistException("The main character has at
least one parent. His name is \"" + parent.toString() + "\"");
            }else{
                parent = parents[0];
                parent.setActivity(true);
                throw new HumanDoesNotExistException("The \"main\" parent of the
main character is " + parents[0].toString());
            }
        }else{
            parent = parent1;
        }
    }
    public void setKarlson(Man.Karlson Karlson) throws
HumanDoesNotExistException {
        if (Karlson == null) {
            karlson = new Man.Karlson("Карлсон, который живет на крыше");
            throw new HumanDoesNotExistException("The name of the character with
```

```
propeller is incorrect: null. The corrected name: \"Карлсон, который живет на
крыше\"");
        }else{
            karlson = Karlson;
        }
    }
    public void setBeloved(Child child, Child[] children) throws
HumanDoesNotExistException {
        if (child == null){
            if (countchildren > 1){
                if (!children[0].equals(themainchild)) {
                    beloved = children[0];
                }else{
                    beloved = children[1];
                }
            }else {
                throw new HumanDoesNotExistException(themainchild.toString() + "
doesn't have brothers and sisters. It's so sad(");
            }
        }else {
            beloved = child;
        }
    }
    public boolean fear(Human[] humans) {
        int countnotfear = 0;
        int countfear = 0;
        for (int i = 0;i < countthieves; i++) {
            if (humans[i].getFear()) {
                countfear++;
            } else {
                countnotfear++;
            }
        }
        return countnotfear > countfear;
    }
    public Child[] makechildren() {
        Child[] makechildren = new Child [10];
        for (int i = 0; i< countfamily1; i++){
            if (family1[i].getStatus() == Status.CHILD) {
                makechildren[countchildren++] = (Child) family1[i];
            }
        }
        return makechildren;
    }
    public void enumerateWithout (Human h, Human[] people, int count) {
        if (count > 2) {
            Human[] l = new Human[--count];
            int k = 0;
            // Список всех, кого надо перечислить
            for (int j = 0; j < count; j++) {
                if (h.equals(people[j])) {
                    l[k] = people[j];
                    k += 1;
                }
            }
            enumerate(l, k);
        }else {
            if (count == 2) {
                if (h.equals(people[0])) {
```

```
                    System.out.print(people[0]);
                } else {
                    System.out.print(people[1]);
                }
            }
        }
    }
    public void enumerate(Human[] people, int count){
        //Само перечисление
        if (count > 1) {
            for (int j = 0; j < count - 2; j++) {
                System.out.print(people[j].toString());
                System.out.print(", ");
            }
            System.out.print(people[count - 2].toString() + " и " + people[count
- 1].toString());
        }else{
            System.out.print(people[0]);
        }
    }
    public boolean checkDog(Human[] humans, int count){
        boolean b = false;
        for (int i = 0; i < count; i++){
            if (humans[i].getStatus() == Status.DOG){
                b = true;
                break;
            }
        }
        return !b;
    }
    public String theend(int time, Human human){
        switch (time){
            case 1:
                if (human.getGender() == Gender.MALE){
                    return "";
                }else{
                    if (human.getGender() == Gender.FEMALE){
                        return "а";
                    }
                    return "и";
                }
            case 2:
                if (human.getGender() == Gender.MALE){
                    return "его";
                }else{
                    if (human.getGender() == Gender.FEMALE){
                        return "ее";
                    }
                    return "их";
                }
            case 3:
                if (human.getGender() == Gender.MALE){
                    return "ся";
                }else{
                    return "ась";
                }
            case 6:
                if (human.getGender() == Gender.MALE){
                    return "";
```

```
                }
            case 4:
                if (human.getGender() == Gender.MALE){
                    return "ел";
                }else{
                    return "ла";
                }
            case 7:
                if (human.getGender() == Gender.MALE){
                    return "ему";
                }
            case 5:
                if (human.getGender() == Gender.MALE){
                    return "им";
                }else{
                    return "ей";
                }
            case 11:
                if (human.getGender() == Gender.MALE){
                    return "ой";
                }
            case 10:
                if (human.getGender() == Gender.MALE){
                    return "ий";
                }else{
                    return "ая";
                }
            default:
                return "";
        }
    }
}
```

7) thieves

a) attributesofthieves

```
package thieves;

public interface attributesofthieves {
    void setNumber(int number);
    String rush();
    String checkFear();
    String drop();
}
```

b) Thief

```
package thieves;

import human.Gender;
import human.Human;
import human.Personality;
import human.Status;

public class Thief extends Human implements attributesofthieves {

    private int number = 0;

    public Thief(){
        setThief();
    }
```

```
public Thief(String name){
    super(name);
    setThief();
}
public Thief(String name, Gender Gender){
    super(name, Gender);
    setThief();
}
public Thief(Gender Gender){
    super(Gender);
    setThief();
}
public Thief(boolean fear){
    this.setFear(fear);
    setThief();
}
public Thief(String name, boolean fear){
    super(name);
    setThief();
    this.setFear(fear);
}
public Thief(String name, Gender Gender, boolean fear){
    super(name, Gender);
    setThief();
    this.setFear(fear);
}
public Thief(Gender Gender, boolean fear){
    super(Gender);
    setThief();
    this.setFear(fear);
}

public void setThief(){
    setPersonality(Personality.BAD);
    setStatus(Status.THIEF);
}


public void setNumber(int number){
    this.number = number;
    if (number > 1){
        this.setGender(Gender.IT);
    }
}

public String rush(){
    if (this.getFear()) {
        if (this.number == 1) {
            if (this.getGender() == Gender.FEMALE) {
                return "бросилась к двери, ";
            } else {
                return "бросился к двери, ";
            }
        }
        return "бросились к двери, ";
    }else{
        if (this.number == 1) {
            if (this.getGender() == Gender.FEMALE) {
                return "пошла к двери, ";
```

```
                } else {
                    return "пошел к двери, ";
                }
            }
            return "пошли к двери, ";
        }
    }

    public String checkFear(){
        if (this.getFear()){
            if (this.number == 1) {
                if (this.getGender() == Gender.FEMALE) {
                    return "Не помня себя от страха, она выскочила в прихожую, а
оттуда на лестничную площадку. ";
                } else {
                    return "Не помня себя от страха, он выскочил в прихожую, а
оттуда на лестничную площадку. ";
                }
            }
            return "Не помня себя от страха, они выскочили в прихожую, а оттуда
на лестничную площадку. ";
        }
        if (this.number == 1) {
            if (this.getGender() == Gender.FEMALE) {
                return "Это ее насторожило и она решили выйти из квартиры как
можно скорее. ";
            } else {
                return "Это его насторожило и он решили выйти из квартиры как
можно скорее. ";
            }
        }
        return "Это их насторожило и они решили выйти из квартиры как можно
скорее. ";
    }

    public String drop(){
        if (this.getGender() == Gender.FEMALE){
            return "которые уронила " + this.toString() + ", когда она металась
между кухней и столовой. ";
        }else{
            return "которые уронил " + this.toString() + ", когда он метался
между кухней и столовой. ";
        }
    }

}
```

Результат работы программы:

```
THIEF Рулле и THIEF Филле бросились к двери, а приведение вилось вокруг них. Не помня себя от страха, они выскочили в прихожую, а оттуда на лестничну
ю площадку. Привидение преследовало по пятам, гнало вниз по лестнице и выкрикивало время от времени глухим, страшным голосом: Но потом привидение уст
ало и вернулось в столовую. CHILD Малыш собрал с пола деньги, кольца, брошки и положил все это обратно в секретер. CHILD Гунилла и CHILD Кристер подо
брали все вилки и ложки, которые уронил THIEF noname, когда он метался между кухней и столовой. Дети смеялись; они были счастливы. CHILD Малыш прыгал
 от радости, что все обернулось так хорошо.
На следующее утро, едва проснувшись, взъерошенный мальчуган в полосатой голубой пижаме пришлепал босиком к PARENT Мама на кухню. PARENT Папа уже ушел
 на службу. CHILD Боссе и CHILD Бетан - в школу. CHILD Малыш обычно вставал раньше, чем надо было, потому что он любил оставаться вот так по утрам вд
воем с PARENT Мама, пусть даже ненадолго. В такие минуты хорошо разговаривать, вместе петь песни или рассказывать друг другу сказки.
Когда CHILD Малыш вошел в кухню, PARENT Мама, примостившись у кухонного стола, пила кофе и читала газету. CHILD Малыш сел рядом с PARENT Мама, так он
и сидели, пока CHILD Малыш окончательно не проснулся.
PARENT Мама и PARENT Папа вернулись вчера с прогулки позже, чем предполагали. CHILD Малыш еще не спал, но было уже очень поздно. Однако PARENT Мама з
аметила дырки, прорезанные в простыне. А сама простыня была такая грязная, словно ее кто-то нарочно исчертил углем. А теперь, когда озорник пришел на
 кухню, она твердо решила не отпускать его без объяснений.
CHILD Малыш молчал и напряженно думал. Как быть? Ведь дырки прорезал именно KARLSON Карлсон, который живет на крыше, а PARENT Мама запретила о нем го
ворить. CHILD Малыш решил также ничего не рассказывать и о ворах, потому что PARENT Мама все равно этому не поверит.
"А! Значит, это CHILD Гунилла разрезала простыню", - подумала PARENT Мама. И еще она подумала, что его CHILD Малыш - хороший мальчик, потому что он н
е желает наговаривать на других, а хочет, чтобы CHILD Гунилла сама все рассказала. PARENT Мама обняла CHILD Малыш за плечи. Она решила сейчас больше
ни о чем его не расспрашивать, но при случае поговорить с CHILD Гунилла. PARENT Мама вновь принялась читать газету,  а CHILD Малыш молча сидел у нее
на коленях и думал. Кого же, собственно говоря, он действительно любит? Прежде всего он любит PARENT Мама...  и PARENT Папа тоже... Еще он любит null
... Ну да, чаще всего он их все-таки любит, особенно CHILD Бетан. Но иногда он на нее так сердится, что вся любовь пропадает.  Любит он и KARLSON Кар
лсон, который живет на крыше, и CHILD Гунилла тоже любит. Да, быть может, он женится на ней, когда вырастет, потому что хочешь не хочешь, а жену имет
ь надо. Конечно, больше всего он хотел бы жениться на маме, но ведь это невозможно. Вдруг CHILD Малыш пришла в голову мысль, которая его встревожила.
 PARENT Мама подвинула к себе чашку и с удивлением взглянула на CHILD Малыш. CHILD Малыш, испугавшись, что сморозил глупость, решил не продолжать. Но
 PARENT Мама настаивала: Да, это было так. Тут PARENT Мама расмеялась и сказала:
- Раз вы оба меня любите, значит я хорошая.
- Ну, тогда я женюсь на CHILD Гунилла, - вздохнул CHILD Малыш. - Ведь надо же мне будет на ком-нибудь жениться!
CHILD Малыш задумался. Он думал о том, что ему, наверное, будет не очень приятно жить вместе с CHILD Гунилла, потому что с ней трудно ладить. Да и во
обще ему больше всего хотелось жить вмсете с PARENT Мама, PARENT Папа и CHILD Бетан, а не с какой-то там женой.
В тот день CHILD Малыш было приятно идти в школу, потому что ему многое надо было обсудить с CHILD Гунилла и CHILD Кристер. Домой они шли, как всегда
, вместе. И CHILD Малыш это особенно радовало, потому что теперь CHILD Гунилла и CHILD Кристер тоже были знакомы с KARLSON Карлсон, который живет на
крыше.
Тут появилось еще одно существо, которое тоже захотело пойти вместе с ними. Когда ребята собрались перейти улицу, к CHILD Малыш подбежал маленький че
рный пудель. Он обнюхал коленки CHILD Малыш и дружески тявкнул. Однако собака была такой большой, что ее появление не обрадовало CHILD Малыш, а даже
несколько испугала. К счастью, рядом была ее DOGOWNER владелица собаки.
Вскоре CHILD Малыш пришел домой. Но PARENT Мама не было дома. В записке, которую CHILD Малыш нашел на кухонном столе, было сказано, что она в прачечн
ой и что он может туда зайти, если ему что-нибудь понадобится. CHILD Малыш вошел в свою комнату. В эту самую минуту в окно влетел KARLSON Карлсон, ко
торый живет на крыше. s311793@helios:/home/s311793/Lab_4$
```

Вывод:

Изучены использование локальных, анонимных и вложенных классов (static и non-static), создание классов исключений (checked и unchecked), а также обработка исключений этих классов.