

Санкт-Петербургский Национальный Исследовательский  
Университет ИТМО  
Мегафакультет Компьютерных Технологий и Управления  
Факультет Программной Инженерии и Компьютерной Техники



**Вариант №311678876767.6**  
**Лабораторная работа №4**  
по дисциплине  
**Программирование**

Выполнил Студент группы Р3116  
**Алексей Лапин**  
Преподаватель:  
**Сорокин Роман Борисович**

г. Санкт-Петербург  
2021г.

# Содержание

1	Текст задания	3
2	Диаграмма классов объектной модели.	4
3	Исходный код программы	4
4	Результат выполнения:	17
5	Вывод	17

# 1 Текст задания

**Программа должна удовлетворять следующим требованиям:**

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

**Порядок выполнения работы:**

- Доработать объектную модель приложения.
- Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
- Согласовать с преподавателем изменения, внесённые в модель.
- Модифицировать программу в соответствии с внесёнными в модель изменениями.

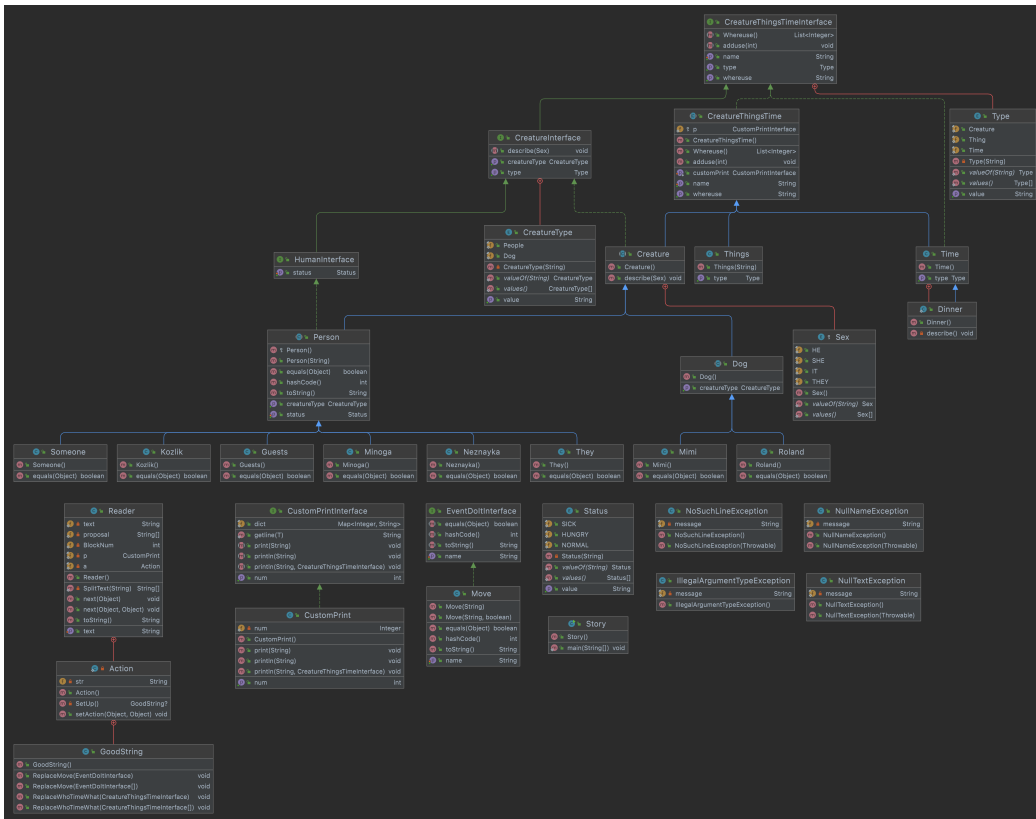
**Описание предметной области, по которой должна быть построена объектная модель:**

Нужно сказать, что Незнайка никогда не забывал о своём больном друге. Не проходило дня, чтоб он не забежал к нему хотя бы на минутку. Обычно это удавалось сделать во время послеобеденной прогулки. Всегда, когда Незнайка обедал с собаками, он не съедал свою порцию до конца, а припрятывал в карман то пирожок, то котлетку, то хлеба краюшку и относил всё это голодному Козлику.

В первый же день он обратился к госпоже Миноге с просьбой заплатить ему жалованье хотя бы за недельку вперёд, так как ему нужно помочь больному приятелю, который находился в дрянингской ночлежке. Госпожа Минога сказала, что теперь он живёт в богатом доме, в обществе приличных собак, и ему не пристало водить компанию с каким-то Козликом, который даже дома собственного не имеет, а обитает в какой-то ночлежке.

– Ни о каких таких Козликах я не желаю и слышать! – сказала она. Если же вы произнесёте при мне или при Мимишке с Роландом какоенибудь неприличное слово вроде «ночлежки», я вас уволю. Что же касается платы, то вы будете получать её раз в неделю, но только не вперёд, а по прошествии недели.

## 2 Диаграмма классов объектной модели.



## 3 Исходный код программы

Story.java

```
1 package core;
2
3 import core.CreatureThingsTime.*;
4 import core.CreatureThingsTime.dog.Dog;
5 import core.CreatureThingsTime.dog.Mimi;
6 import core.CreatureThingsTime.dog.Roland;
7 import core.CreatureThingsTime.people.*;
8 import core.event.*;
9 import core.print.CustomPrintInterface;
10 import core.reader.Reader;
11
12 import java.util.Random;
13
14 public class Story {
15     public static void main(String[] args) {
16         Reader r = new Reader();
17         CreatureThingsTimeInterface din = new Time.Dinner();
18         r.next(new Move("приходила"), din);
19         r.next(new Move("проводили"));
20         HumanInterface mi = new Minoga();
21         r.next(new Move("был"), new CreatureThingsTimeInterface[]{mi, new Things("званный_
22             вечер")});
23         HumanInterface nez = new Nezmayka();
24         CreatureInterface mimi = new Mimi();
25         CreatureInterface rol = new Roland();
26         r.next(new Move("приводил"), new CreatureThingsTimeInterface[]{nez, mimi, rol});
27         HumanInterface guests = new Guests();
28         CreatureThingsTime dogs = new Dog() {
29             {
30                 super.name = "собаками";
31                 super.describe(Sex.THEY);
32             }
33         }
```

```

32     };
33     r.next(new Move("могли_полюбоваться"), new CreatureThingsTimeInterface[]{guests,dogs
    });
34     r.next(new Move("уходила"),new CreatureThingsTimeInterface[]{mi, new Things("в_
    tearp"))});
35     r.next(new Move("брала"), mimi);
36     r.next(new Move[]{new Move("была"),new Move("таскать")});
37     HumanInterface who = new Someone();
38     r.next(new Move("являлся"),who);
39     r.next(new Move[]{new Move ("считали"), new Move("смеялись")});
40     r.next(new Move("оставался"),new CreatureThingsTimeInterface[]{nez,rol});
41     HumanInterface they = new They();
42     r.next(new Move("отправлялись"), they);
43     r.next(new Move("_смотрели"));
44     HumanInterface koz = new Kozlik();
45     koz.setStatus(Status.SICK);
46     r.next(new Move[]{new Move("отправлялись"), new Move("навещали")},koz);
47     r.next(new Move("Нужно_сказать"));
48     r.next(new Move("забывал",false),nez);
49     r.next(new Move("проходило",false),new Things("дня"));
50     HumanInterface he = new Person() {
51         {
52             super.setName("он");
53             super.describe(Sex.HE);
54         }
55     };
56     r.next(new Move("забежал",false),he);
57     r.next(new Move("удавалось_сделать"),new Things("это"));
58     r.next(new Move("обедал",nez);
59     r.next(new Move("съедал",false),he);
60     r.next(new Move("припрятывал"),new Things("пирожок"));
61     r.next(new Things("котлетку"));
62     koz.setStatus(Status.HUNGRY);
63     r.next(new Move("относил"),new CreatureThingsTimeInterface[]{new Things("хлебца_
    краюшку"),koz});
64     r.next(new Move[]{new Move("обратился"), new Move("с_просьбой_заплатить")},
    new CreatureThingsTimeInterface[]{nez,mi});
65     r.next(new Move("нужно_помочь"));
66     HumanInterface which = new Person() {
67         {
68             super.setName("который");
69             super.describe(Sex.HE);
70         }
71     };
72     r.next(new Move("находился"), which);
73     r.next(new Move("сказала"),mi);
74     r.next(new Move("живёт"),he);
75     r.next(new Move("пристало",false),koz);
76     r.next(new Move("не_имеет"),which);
77     r.next(new Move("обитает"));
78     HumanInterface I = new Person() {
79         {
80             super.setName("я");
81             super.describe(Sex.HE);
82         }
83     };
84     r.next(new Move[]{new Move("желаю",false),new Move("слышать")},
    new CreatureThingsTimeInterface[]{koz,I});
85     HumanInterface she = new Person() {
86         {
87             super.setName("она");
88             super.describe(Sex.SHE);
89         }
90     };
91     r.next(new Move("сказала"),she);
92     HumanInterface you = new Person() {
93         {
94             super.setName("вы");
95             super.describe(Sex.SHE);
96         }
97     };
98     r.next(new Move("произнесёте"),new CreatureThingsTimeInterface[]{you,mimi,rol});
99     r.next(new Move("уволю"),I);
100    r.next(new Move("будете_получать"),you);
101    r.next(null,null);
102
103    System.out.println(mi.getWhereuse());
104
105

```

```

106     System.out.println(nez.getWhereuse());
107     System.out.println(koz.getWhereuse());
108     Integer[] ar = new Integer[3];
109     for (int i = 0; i < ar.length; i++){
110         ar[i] = new Random().nextInt(61);
111     }
112     System.out.println(CustomPrintInterface.getline(ar));
113
114 }
115 }

```

reader/Reader.java

```

1  package core.reader;
2
3  import core.CreatureThingsTime.*;
4  import core.event.EventDoItInterface;
5  import core.exception.IllegalArgumentException;
6  import core.exception.NullTextException;
7  import core.print.CustomPrint;
8  import core.print.CustomPrintInterface;
9
10
11 import java.util.Arrays;
12
13 public class Reader {
14     private static String text = "Наконец_move_time," +
15         "после_которого_время_move_поразному-." +
16         "Если_у_who_move_what," +
17         "то_who_move_who_и_who_в_комнаты," +
18         "чтоб_who_move_who." +
19         "Если_who_move_what," +
20         "то_обязательно_move_с_собой_и_who," +
21         "потому_что_в_то_время_move_мода_move_по_театрам_своих_комнатных_собачонок." +
22         "Всех,_who_move_в_театр_или_на_концерт_без_собаки," +
23         "move_неимущими_бедняками_и_move_над_ними." +
24         "В_такие_вечера_на_попеченье_who_move_один_who," +
25         "и_who_move_вдвоём_в_спортивный_собачий_зал_или_плавательный_бассейн," +
26         "где_move_какое_нибудь_собачье_состязание," +
27         "или_же_move_в_дрянный_«Тупичок»_и_move_большого_who." +
28         "move,_что_who_никогда_move_о_своём_большом_друге." +
29         "move_what," +
30         "чтоб_who_move_к_нему_хотя_бы_на_минутку." +
31         "Обычно_what_move_во_время_послеобеденной_прогулки." +
32         "Всегда," +
33         "когда_who_move_с_собаками," +
34         "who_move_свою_порцию_до_конца," +
35         "а_move_в_карман_то_what," +
36         "то_what," +
37         "то_what_и_move_всё_это_голодному_who." +
38         "В_первый_же_день_who_move_к_who_move_ему_жалованье_хотя_бы_за_недельку_вперёд," +
39         "так_как_ему_move_большому_приятелю," +
40         "who_move_в_дрянной_ночлежке." +
41         "who_move," +
42         "что_теперь_who_move_в_богатом_доме," +
43         "в_обществе_приличных_собак," +
44         "и_ему_move_компанию_с_какимто_who," +
45         "who_даже_дома_собственного_move," +
46         "а_move_в_какойто_ночлежке." +
47         "Ни_о_каких_таких_who_who_move_и_move!" +
48         "move_who." +
49         "Если_же_who_move_при_мне_или_при_who_с_who_какое_нибудь_неприличное_слово_вроде_
50         «ночлежки»," +
51         "who_вас_move." +
52         "Что_же_касается_платы," +
53         "то_who_move_её_раз_в_неделю," +
54         "но_только_не_вперёд,_а_до_прошествии_недели.";
55     private static String[] proposal;
56     private static int BlockNum = 0;
57     private static final CustomPrint p = new CustomPrint();
58     private static final Action a = new Action();
59
60     public Reader(){
61         proposal = SplitText(text);
62         CreatureThingsTime.setCustomPrint(p);
63     }
64
65     public void next(Object obj1){

```

```

65     Object obj2 = null;
66     if (obj1 instanceof EventDoItInterface | obj1 instanceof EventDoItInterface[]) {
67         a.setAction(obj1, obj2);
68     } else {
69         a.setAction(obj2, obj1);
70     }
71 }
72 public void next(Object act, Object cre){
73     a.setAction(act, cre);
74 }
75
76 private static String [] SplitText(String text){
77     return text.split("(?<=[.!,?])[\s-]");
78 }
79
80 public void setText(String text) {
81     if (text.equals("")){
82         throw new NullTextException();
83     }
84     else{Reader.text = text;}
85 }
86
87 @Override
88 public String toString() {
89     return "Reader{" +
90         "text='" + text + '\'' +
91         ", proposal=" + Arrays.toString(proposal) +
92         '}';
93 }
94 private static class Action {
95     private String str;
96     public void setAction(Object act, Object cre){
97         GoodString a = SetUp();
98         if (a != null) {
99             if (act instanceof EventDoItInterface[]) {
100                 a.ReplaceMove((EventDoItInterface[]) act);
101             } else if (act instanceof EventDoItInterface) {
102                 a.ReplaceMove((EventDoItInterface) act);
103             } else {
104                 if (act != null) {
105                     throw new IllegalArgumentException();
106                 }
107             }
108             if (cre instanceof CreatureThingsTimeInterface[]) {
109                 a.ReplaceWhoTimeWhat((CreatureThingsTimeInterface[]) cre);
110             } else if (cre instanceof CreatureThingsTimeInterface) {
111                 a.ReplaceWhoTimeWhat((CreatureThingsTimeInterface) cre);
112             } else {
113                 if (cre != null) {
114                     throw new IllegalArgumentException();
115                 }
116             }
117             p.println(str);
118         }
119         else p.println("to be continued...");
120     }
121
122     private GoodString SetUp(){
123         while (true) {
124             if (BlockNum==proposal.length){
125                 return null;
126             }
127             str = proposal[BlockNum];
128             if (str.matches(".*(\\w)+.*")) {
129                 GoodString gs = new GoodString();
130                 BlockNum = BlockNum + 1;
131                 return gs;
132             }
133             p.println(str);
134             BlockNum = BlockNum + 1;
135         }
136     }
137     public class GoodString{
138         public void ReplaceMove(EventDoItInterface[] act){
139             if (str.matches(".*(move)+.*")) {
140                 for (EventDoItInterface a : act) {
141                     str = str.replaceFirst("move", a.getName());

```

```

142     }
143     }
144 }
145 public void ReplaceMove(EventDoItInterface act){
146     if (str.matches(".*(move)+.*")) {
147         str = str.replaceFirst("move", act.getName());
148     }
149 }
150 public void ReplaceWhoTimeWhat(CreatureThingsTimeInterface[] cre){
151     if (str.matches(".*(who|time|what)+.*")) {
152         for (CreatureThingsTimeInterface c : cre) {
153             if (c instanceof CreatureInterface) {
154                 str = str.replaceFirst("who", c.getName());
155             }
156             else if (c instanceof Time){
157                 str = str.replaceFirst("time", c.getName());
158             }
159             else {
160                 str = str.replaceFirst("what", c.getName());
161             }
162             c.adduse(p.getNum());
163         }
164     }
165 }
166 public void ReplaceWhoTimeWhat(CreatureThingsTimeInterface cre){
167     if (str.matches(".*(who|time|what)+.*")) {
168         if (cre instanceof CreatureInterface) {
169             str = str.replaceFirst("who", cre.getName());
170         }
171         else if (cre instanceof Time){
172             str = str.replaceFirst("time", cre.getName());
173         }
174         else {
175             str = str.replaceFirst("what", cre.getName());
176         }
177         cre.adduse(p.getNum());
178     }
179 }
180 }
181 }
182 }

```

print/CustomPrint.java

```

1 package core.print;
2
3 import core.CreatureThingsTime.CreatureThingsTimeInterface;
4
5 public class CustomPrint implements CustomPrintInterface{
6     private static Integer num = 0;
7     public void print(String str){
8         System.out.print(num+"."+str);
9         dict.put(num, str);
10        num+=1;
11    }
12    public void println(String str){
13        System.out.print(num+"."+str+"\n");
14        dict.put(num, str);
15        num+=1;
16    }
17    public void println(String str, CreatureThingsTimeInterface cre){
18        cre.adduse(num);
19        System.out.print(num+"."+str+"\n");
20        dict.put(num, str);
21        num+=1;
22    }
23
24    @Override
25    public int getNum() {
26        return num;
27    }
28 }

```

print/CustomPrintInterface.java

```

1 package core.print;
2
3

```



```

4 import core.CreatureThingsTime.CreatureThingsTimeInterface;
5 import core.exception.IllegalArgumentException;
6 import core.exception.NoSuchLineException;
7
8 import java.util.HashMap;
9 import java.util.Map;
10
11 public interface CustomPrintInterface {
12     Map<Integer,String> dict = new HashMap<>();
13     public void print(String str);
14     public void println(String str);
15     public void println(String str, CreatureThingsTimeInterface cre);
16     public int getNum();
17     public static<T> String getline(T num) {
18         if (num instanceof Integer) {
19             try {
20                 return "Строчка_" + num + ":_ " + dict.get(num);
21             } catch (RuntimeException e) {
22                 throw new NoSuchLineException(e);
23             }
24         }
25         else if (num instanceof Integer[]){
26             String str="";
27             for (int i : (Integer[])num) {
28                 try {
29                     str+= "Строчка_" + i + ":_ " + dict.get(i)+"\n";
30                 } catch (RuntimeException e) {
31                     throw new NoSuchLineException(e);
32                 }
33             }
34             return str.substring(0, str.length() - 1);
35         }
36         else throw new IllegalArgumentException();
37     };
38 }

```

exception

IllegalArgumentException.java

```

1 package core.exception;
2
3 public class IllegalArgumentException extends RuntimeException{
4     private static final String message = "Передан_не_правильный_тип_аргумента";
5
6     public IllegalArgumentException() {
7         super(message);
8     }
9 }

```

NoSuchLineException.java

```

1 package core.exception;
2
3 public class NoSuchLineException extends RuntimeException{
4     private static final String message = "Строка_не_найдена!";
5
6     public NoSuchLineException() {
7         super(message);
8     }
9
10    public NoSuchLineException(Throwable cause) {
11        super(message, cause);
12    }
13 }

```

NullNameException.java

```

1 package core.exception;
2
3 public class NullNameException extends Exception{
4     private static final String message = "Error!_Name_can't_be_empty";
5
6
7     public NullNameException() {
8         super(message);
9     }
10 }

```

```

11     public NullNameException(Throwable cause) {
12         super(message, cause);
13     }
14
15 }

```

#### NullTextException.java

```

1  package core.exception;
2
3  public class NullTextException extends RuntimeException{
4      private static final String message = "Текст не может быть пустым!";
5
6
7      public NullTextException() {
8          super(message);
9      }
10
11     public NullTextException(Throwable cause) {
12         super(message, cause);
13     }
14 }

```

#### event

#### EventDoItInterface.java

```

1  package core.event;
2
3  public interface EventDoItInterface {
4      public void setName(String act);
5      public String getName();
6      public String toString();
7      public boolean equals(Object obj);
8      public int hashCode();
9  }

```

#### Move.java

```

1  package core.event;
2
3  public class Move implements EventDoItInterface {
4      private String name;
5      public Move(String name){
6          this.name=name;
7      }
8
9      public Move(String name, boolean do_it){
10         if(!do_it){
11             this.name="не "+name;
12         }
13         else this.name=name;
14     }
15
16
17     @Override
18     public void setName(String name) {
19         this.name = name;
20     }
21
22     @Override
23     public String getName() {
24         return name;
25     }
26
27     @Override
28     public String toString() {
29         return "Move{" +
30             "name='" + name + '\'' +
31             '}';
32     }
33
34     @Override
35     public boolean equals(Object obj){
36         if (this == obj) return true;
37         if(obj instanceof Move){
38             return true;
39         }

```

```

40         return false;
41     }
42     @Override
43     public int hashCode(){
44         return this.getName() == null ? 0 : this.getName().hashCode();
45     }
46 }

```

## CreatureThingsTime

### Creature.java

```

1  package core.CreatureThingsTime;
2
3  import core.print.CustomPrintInterface;
4
5  public abstract class Creature extends CreatureThingsTime implements CreatureInterface{
6
7      public void describe(Creature.Sex s){
8          class Describe{
9              private final CustomPrintInterface p;
10             private final Sex s;
11             private final CreatureThingsTimeInterface cre;
12
13             public Describe(CustomPrintInterface p,Creature.Sex s,
14                 CreatureThingsTimeInterface cre){
15                 this.p = p;
16                 this.s = s;
17                 this.cre = cre;
18             }
19             public void sexDescribe(){
20                 switch (s){
21                     case HE -> {
22                         p.println("На сцене появился " + name,cre);
23                     }
24                     case SHE -> {
25                         p.println("На сцене появилась " + name,cre);
26                     }
27                     case IT -> {
28                         p.println("На сцене появилось " + name,cre);
29                     }
30                     case THEY -> {
31                         p.println("На сцене появились " + name,cre);
32                     }
33                 }
34             }
35             Describe d = new Describe(p,s,this);
36             d.sexDescribe();
37         }
38
39         protected enum Sex{
40             HE,
41             SHE,
42             IT,
43             THEY
44         }
45     }

```

### CreatureInterface.java

```

1  package core.CreatureThingsTime;
2
3  public interface CreatureInterface extends CreatureThingsTimeInterface {
4      public void describe(Creature.Sex s);
5      @Override
6      default Type getType() {
7          return Type.Creature;
8      }
9      public CreatureType getCreatureType();
10     public enum CreatureType{
11         People("Человек"),
12         Dog("Собака");
13         private final String value;
14
15         CreatureType(String value) {
16             this.value = value;
17         }

```

```

18
19         public String getValue() {
20             return value;
21         }
22     }
23 }

```

#### CreatureThingsTime.java

```

1  package core.CreatureThingsTime;
2
3  import core.exception.NullNameException;
4  import core.print.CustomPrintInterface;
5
6  import java.util.ArrayList;
7  import java.util.List;
8
9  public abstract class CreatureThingsTime implements CreatureThingsTimeInterface {
10     protected String name;
11     protected List<Integer> whereuse = new ArrayList<Integer>();
12     protected static CustomPrintInterface p;
13     public void setName(String name){
14         try{
15             if (!(name.equals(""))){
16                 this.name = name;
17             }
18             else{
19                 throw new NullNameException();
20             }
21         }
22         catch (NullNameException nne){
23             System.err.println(nne.getMessage());
24             nne.printStackTrace();
25         }
26     }
27
28
29     public String getName(){
30         return name;
31     }
32     public static void setCustomPrint(CustomPrintInterface p){
33         CreatureThingsTime.p = p;
34     }
35     public void adduse(int numuse) {
36         whereuse.add(numuse);
37     }
38
39     public List<Integer> Whereuse() {
40         return whereuse;
41     }
42     public String getWhereuse(){
43         StringBuilder str = new StringBuilder(name + " встречается в строках");
44         str.append(Whereuse());
45         return str.toString();
46     }
47 }

```

#### CreatureThingsTimeInterface.java

```

1  package core.CreatureThingsTime;
2
3  import core.print.CustomPrintInterface;
4
5  import java.util.ArrayList;
6  import java.util.List;
7
8  public interface CreatureThingsTimeInterface {
9     public void setName(String name);
10     public String getName();
11     public Type getType();
12     public void adduse(int numuse);
13     public List<Integer> Whereuse();
14     public String getWhereuse();
15     public enum Type{
16         Creature("Существо"),
17         Thing("Вещь"),
18         Time("Время");
19         private final String value;

```

```

20
21     Type(String value) {
22         this.value = value;
23     }
24
25     public String getValue() {
26         return value;
27     }
28 }
29 }

```

#### Status.java

```

1  package core.CreatureThingsTime;
2
3  public enum Status{
4      SICK("болен"), HUNGRY("голоден"), NORMAL("нормальном");
5      private final String value;
6
7      Status(String value) {
8          this.value = value;
9      }
10
11     public String getValue() {
12         return value;
13     }
14 }

```

#### Things.java

```

1  package core.CreatureThingsTime;
2
3  public class Things extends CreatureThingsTime{
4      public Things(String name){
5          super.name=name;
6      }
7      @Override
8      public Type getType() {
9          return Type.Thing;
10     }
11 }

```

#### Time.java

```

1  package core.CreatureThingsTime;
2
3  public class Time extends CreatureThingsTime implements CreatureThingsTimeInterface{
4
5      @Override
6      public Type getType() {
7          return Type.Time;
8      }
9      public static class Dinner extends Time{
10         public Dinner(){
11             super.name = "пора_ужина";
12             this.describe();
13         }
14         private void describe(){
15             p.println("Время_ужина");
16         }
17     }
18 }

```

#### CreatureThingsTime/people

##### Guests.java

```

1  package core.CreatureThingsTime.people;
2
3  public class Guests extends Person{
4      public Guests() {
5          super.setName("гости");
6          super.describe(Sex.THEY);
7      }
8
9      @Override
10     public boolean equals(Object obj){
11         if (this == obj) return true;

```

```

12         if(obj instanceof Guests){
13             return true;
14         }
15         return false;
16     }
17 }

```

#### HumanInterface.java

```

1 package core.CreatureThingsTime.people;
2
3 import core.CreatureThingsTime.CreatureInterface;
4 import core.CreatureThingsTime.Status;
5
6
7 public interface HumanInterface extends CreatureInterface {
8
9     public void setStatus(Status status);
10
11     public Status getStatus();
12
13
14
15
16
17
18 }

```

#### Kozlik.java

```

1 package core.CreatureThingsTime.people;
2
3 public class Kozlik extends Person {
4     public Kozlik() {
5         super.setName("Козлик");
6         super.describe(Sex.HE);
7     }
8
9     @Override
10    public boolean equals(Object obj){
11        if (this == obj) return true;
12        if(obj instanceof Kozlik){
13            return true;
14        }
15        return false;
16    }
17 }

```

#### Minoga.java

```

1 package core.CreatureThingsTime.people;
2
3 public class Minoga extends Person {
4     public Minoga() {
5         super.setName("Госпожа Минога");
6         super.describe(Sex.SHE);
7     }
8
9     @Override
10    public boolean equals(Object obj){
11        if (this == obj) return true;
12        if(obj instanceof Minoga){
13            return true;
14        }
15        return false;
16    }
17 }

```

#### Neznayka.java

```

1 package core.CreatureThingsTime.people;
2
3 public class Neznayka extends Person {
4     public Neznayka() {
5         super("Незнайка");
6         super.describe(Sex.HE);
7     }
8

```

```

9      @Override
10     public boolean equals(Object obj){
11         if (this == obj) return true;
12         if(obj instanceof Neznayka){
13             return true;
14         }
15         return false;
16     }
17 }

```

Person.java

```

1  package core.CreatureThingsTime.people;
2
3
4  import core.CreatureThingsTime.Creature;
5  import core.CreatureThingsTime.CreatureThingsTimeInterface;
6  import core.CreatureThingsTime.Status;
7  import core.print.CustomPrintInterface;
8
9
10 public abstract class Person extends Creature implements HumanInterface {
11     private Status status = Status.NORMAL;
12     protected Person(){}
13     public Person(String name){
14         super.name = name;
15     }
16
17     public void setStatus(Status status){
18         this.status = status;
19         p.println("Статус персонажа" + this.getName() + " изменён на" + status.getValue());
20     }
21
22     public Status getStatus(){
23         return this.status;
24     }
25
26     @Override
27     public CreatureType getCreatureType() {
28         return CreatureType.People;
29     }
30
31     @Override
32     public String toString(){
33         return name+" "+status;
34     }
35
36     @Override
37     public boolean equals(Object obj){
38         if (this == obj) return true;
39         if(obj instanceof Person){
40             return true;
41         }
42         return false;
43     }
44     @Override
45     public int hashCode(){
46         int result = name == null ? 0 : name.hashCode();
47         result+=status.hashCode();
48         return result;
49     }
50 }

```

Someone.java

```

1  package core.CreatureThingsTime.people;
2
3  public class Someone extends Person{
4      public Someone() {
5          super("кто");
6          super.describe(Sex.HE);
7      }
8
9      @Override
10     public boolean equals(Object obj){
11         if (this == obj) return true;
12         if(obj instanceof Neznayka){
13             return true;

```

```

14     }
15     return false;
16 }
17 }

```

#### They.java

```

1 package core.CreatureThingsTime.people;
2
3 public class They extends Person{
4     public They() {
5         super.setName("они");
6         super.describe(Sex.THEY);
7     }
8
9     @Override
10    public boolean equals(Object obj){
11        if (this == obj) return true;
12        if(obj instanceof Minoga){
13            return true;
14        }
15        return false;
16    }
17 }

```

#### CreatureThingsTime/dog

##### Dog.java

```

1 package core.CreatureThingsTime.dog;
2
3 import core.CreatureThingsTime.Creature;
4
5 public abstract class Dog extends Creature {
6     @Override
7     public CreatureType getCreatureType() {
8         return CreatureType.Dog;
9     }
10 }

```

##### Mimi.java

```

1 package core.CreatureThingsTime.dog;
2
3 public class Mimi extends Dog{
4     public Mimi() {
5         super.setName("Мими");
6         super.describe(Sex.SHE);
7     }
8
9
10    @Override
11    public boolean equals(Object obj){
12        if (this == obj) return true;
13        if(obj instanceof Mimi){
14            return true;
15        }
16        return false;
17    }
18 }

```

##### Roland.java

```

1 package core.CreatureThingsTime.dog;
2 public class Roland extends Dog {
3     public Roland() {
4         super.setName("Роланд");
5         super.describe(Sex.HE);
6     }
7
8
9     @Override
10    public boolean equals(Object obj){
11        if (this == obj) return true;
12        if(obj instanceof Roland){
13            return true;
14        }
15        return false;

```



```
16     }  
17 }
```

## 4 Результат выполнения:

```
0. Время ужина  
1. Наконец приходила пора ужина,  
2. после которого время проводили по-разному.  
3. На сцене появилась Госпожа Минога  
4. Если у Госпожа Минога был званый вечер,  
5. На сцене появился Незнайка  
6. На сцене появилась Мими  
7. На сцене появился Роланд  
8. то Незнайка приводил Мими и Роланд в комнаты,  
9. На сцене появились гости  
10. На сцене появились собаками  
11. чтоб гости могли полюбоваться собаками.  
12. Если Госпожа Минога уходила в театр,  
13. то обязательно брала с собой и Мими,  
14. потому что в то время была мода таскать по театрам своих комнатных собачонок.  
15. На сцене появился кто  
16. Всех,  
17. кто являлся в театр или на концерт без собаки,  
18. считали неумишми бедняками и смеялись над ними.  
19. В такие вечера на попечение Незнайка оставался один Роланд,  
20. На сцене появились они  
21. и они отправлялись вдвоём в спортивный собачий зал или плавательный бассейн,  
22. где смотрели какое-нибудь собачье состязание,  
23. На сцене появился Козлик  
24. Статус персонажа Козлик изменён на болен  
25. или же отправлялись в дрянингский «Тупичок» и навещали больного Козлик.  
26. Нужно сказать,  
27. что Незнайка никогда не забывал о своём больном друге.  
28. не проходило дня,  
29. На сцене появился он  
30. чтоб он не забежал к нему хотя бы на минутку.  
31. Обычно это удавалось сделать во время послеобеденной прогулки.  
32. Всегда,  
33. когда Незнайка обедал с собаками,  
34. он не съедал свою порцию до конца,  
35. а припрятывал в карман то пирожок,  
36. то котлетку,  
37. Статус персонажа Козлик изменён на голоден  
38. то хлеба краюшку и относил всё это голодному Козлик.  
39. В первый же день Незнайка обратился к Госпожа Минога с просьбой заплатить ему жалованье хотя бы за недельку вперёд,  
40. так как ему нужно помочь больному приятелю,  
41. На сцене появился который  
42. который находился в дрянингской ночлежке.  
43. Госпожа Минога сказала,  
44. что теперь он живёт в богатом доме,  
45. в обществе приличных собак,  
46. и ему не пристало компанию с каким-то Козлик,  
47. который даже дома собственного не имеет,  
48. а обитает в какой-то ночлежке.  
49. На сцене появился я  
50. - Ни о каких таких Козлик я не желаю и слышать!  
51. На сцене появилась она  
52. - сказала она.  
53. На сцене появилась вы  
54. Если же вы произнесёте при мне или при Мими с Роланд какое-нибудь неприличное слово вроде «ночлежки»,  
55. я вас уволю.  
56. Что же касается платы,  
57. то вы будете получать её раз в неделю,  
58. но только не вперёд,  
59. а по прошествии недели.  
60. to be continued...  
Госпожа Минога встречается в строчках [3, 4, 12, 39, 43]  
Незнайка встречается в строчках [5, 8, 19, 27, 33, 39]  
Козлик встречается в строчках [23, 25, 38, 46, 50]  
Строчка 47: который даже дома собственного не имеет,  
Строчка 51: На сцене появилась она  
Строчка 57: то вы будете получать её раз в неделю,
```

## 5 Вывод

В процессе выполнения этой лабораторной работы я познакомился с исключениями в java. Научился использовать локальные, анонимные и вложенные классы.