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 number
Student's name
Преподаватель:
Teacher's name

г. Санкт-Петербург
2021г.

Содержание

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:

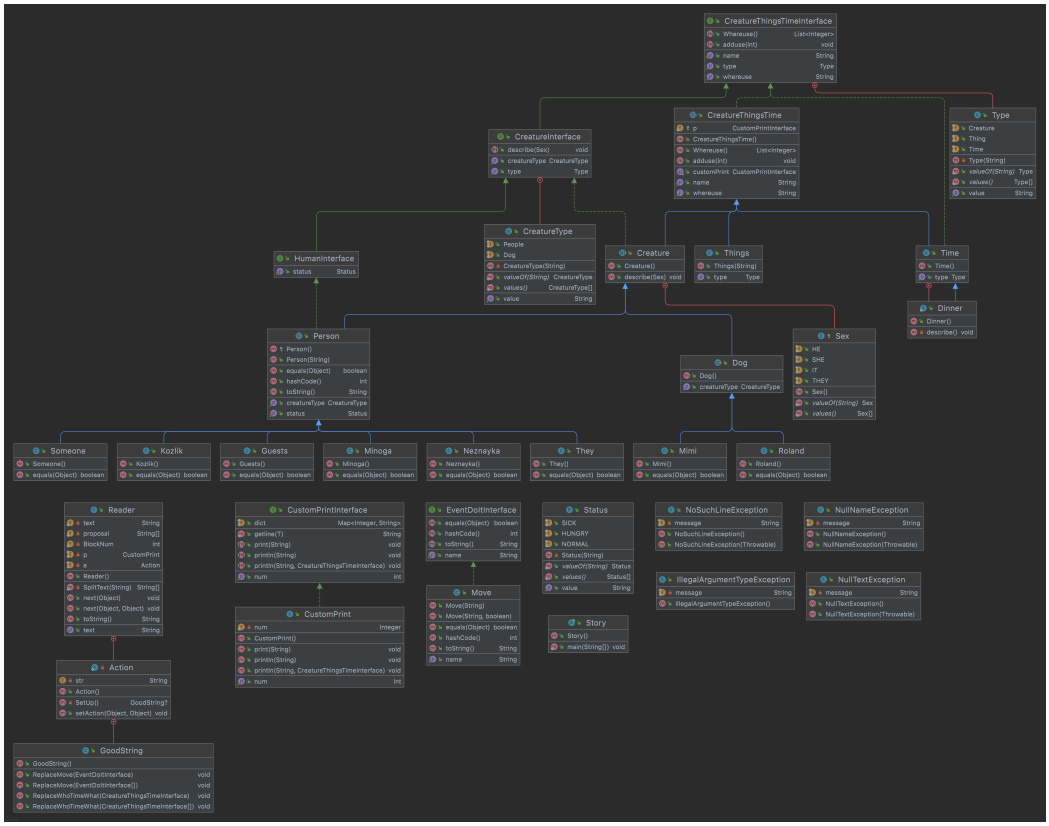
- Доработать объектную модель приложения.
- Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
- Согласовать с преподавателем изменения, внесённые в модель.
- Модифицировать программу в соответствии с внесёнными в модель изменениями.

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 хлебца 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("званый␣
               вечер")});
22         HumanInterface nez = new Neznayka();
23         CreatureInterface mimi = new Mimi();
24         CreatureInterface rol = new Roland();
25         r.next(new Move("приводил"), new CreatureThingsTimeInterface[]{nez, mimi,rol});
26         HumanInterface guests = new Guests();
27         CreatureThingsTime dogs=new Dog(){
28             {
29                 super.name = "собаками";
30                 super.describe(Sex.THEY);
31             }
```

```
32          };
33          r.next(new Move("могли полюбоваться"), new CreatureThingsTimeInterface[]{guests,dogs
                });
34          r.next(new Move("уходила"),new CreatureThingsTimeInterface[]{mi, new Things("в
                театр")});
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")},
65                  new CreatureThingsTimeInterface[]{nez,mi});
66          r.next(new Move("нужно помочь"));
67          HumanInterface which = new Person() {
68              {
69                  super.setName("который");
70                  super.describe(Sex.HE);
71              }
72          };
73          r.next(new Move("находился"), which);
74          r.next(new Move("сказала"),mi);
75          r.next(new Move("живёт"),he);
76          r.next(new Move("пристало",false),koz);
77          r.next(new Move("не имеет"),which);
78          r.next(new Move("обитает"));
79          HumanInterface I = new Person() {
80              {
81                  super.setName("я");
82                  super.describe(Sex.HE);
83              }
84          };
85          r.next(new Move[]{new Move("желаю",false),new Move("слышать")},
86                  new CreatureThingsTimeInterface[]{koz,I});
87          HumanInterface she = new Person() {
88              {
89                  super.setName("она");
90                  super.describe(Sex.SHE);
91              }
92          };
93          r.next(new Move("сказала"),she);
94          HumanInterface you = new Person() {
95              {
96                  super.setName("вы");
97                  super.describe(Sex.SHE);
98              }
99          };
100         r.next(new Move("произнесёте"),new CreatureThingsTimeInterface[]{you,mimi,rol});
101         r.next(new Move("уволю"),I);
102         r.next(new Move("будете получать"),you);
103         r.next(null,null);
104
105         System.out.println(mi.getWhereuse());
```

```
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.IllegalArgumentTypeException;
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 какоенибудь- собачье состязание," +
27             " или же move в дрянингский «Тупичок» и move больного who." +
28             " move, что who никогда move о своём больном 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 больному приятелю," +
40             " who move в дрянингской 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 какоенибудь- неприличное слово вроде
                   «ночлежки»," +
50             " who вас move." +
51             " Что же касается платы," +
52             " то who move её раз в неделю," +
53             " но только не вперёд, а по прошествии недели.";
54     private static String[] proposal;
55     private static int BlockNum = 0;
56     private static final CustomPrint p = new CustomPrint();
57     private static final Action a = new Action();
58 
59     public Reader(){
60        proposal = SplitText(text);
61        CreatureThingsTime.setCustomPrint(p);
62     }
63 
64     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 IllegalArgumentTypeException();
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 IllegalArgumentTypeException();
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());
```

```
                    }
                }
            }
            public void ReplaceMove(EventDoItInterface act){
                if (str.matches(".*(move)+.*")) {
                    str = str.replaceFirst("move", act.getName());
                }
            }
            public void ReplaceWhoTimeWhat(CreatureThingsTimeInterface[] cre){
                if (str.matches(".*(who|time|what)+.*")) {
                    for (CreatureThingsTimeInterface c : cre) {
                        if (c instanceof CreatureInterface) {
                            str = str.replaceFirst("who", c.getName());
                        }
                        else if (c instanceof Time){
                            str = str.replaceFirst("time", c.getName());
                        }
                        else {
                            str = str.replaceFirst("what", c.getName());
                        }
                        c.adduse(p.getNum());
                    }
                }
            }
            public void ReplaceWhoTimeWhat(CreatureThingsTimeInterface cre){
                if (str.matches(".*(who|time|what)+.*")) {
                    if (cre instanceof CreatureInterface) {
                        str = str.replaceFirst("who", cre.getName());
                    }
                    else if (cre instanceof Time){
                        str = str.replaceFirst("time", cre.getName());
                    }
                    else {
                        str = str.replaceFirst("what", cre.getName());
                    }
                    cre.adduse(p.getNum());
                }
            }
        }
    }
}
```

print/CustomPrint.java

```
package core.print;

import core.CreatureThingsTime.CreatureThingsTimeInterface;

public class CustomPrint implements CustomPrintInterface{
    private static Integer num = 0;
    public void print(String str){
        System.out.print(num+". "+str);
        dict.put(num, str);
        num+=1;
    }
    public void println(String str){
        System.out.print(num+". "+str+"\n");
        dict.put(num, str);
        num+=1;
    }
    public void println(String str, CreatureThingsTimeInterface cre){
        cre.adduse(num);
        System.out.print(num+". "+str+"\n");
        dict.put(num, str);
        num+=1;
    }

    @Override
    public int getNum() {
        return num;
    }
}
```

print/CustomPrintInterface.java

```
package core.print;


```

```java
import core.CreatureThingsTime.CreatureThingsTimeInterface;
import core.exception.IllegalArgumentTypeException;
import core.exception.NoSuchLineException;

import java.util.HashMap;
import java.util.Map;

public interface CustomPrintInterface {
    Map<Integer,String> dict = new HashMap<>();
    public void print(String str);
    public void println(String str);
    public void println(String str, CreatureThingsTimeInterface cre);
    public int getNum();
    public static<T> String getline(T num) {
        if (num instanceof Integer) {
            try {
                return "Строчка " + num + ": " + dict.get(num);
            } catch (RuntimeException e) {
                throw new NoSuchLineException(e);
            }
        }
        else if (num instanceof Integer[]){
            String str="";
            for (int i : (Integer[])num) {
                try {
                    str+= "Строчка " + i + ": " + dict.get(i)+"\n";
                } catch (RuntimeException e) {
                    throw new NoSuchLineException(e);
                }
            }
            return str.substring(0, str.length() - 1);
        }
        else throw new IllegalArgumentTypeException();
    };
}
```

exception
IllegalArgumentTypeException.java

```java
package core.exception;

public class IllegalArgumentTypeException extends RuntimeException{
    private static final String message = "Передан не правильный тип аргумента";

    public IllegalArgumentTypeException() {
        super(message);
    }
}
```

NoSuchLineException.java

```java
package core.exception;

public class NoSuchLineException extends RuntimeException{
    private static final String message = "Строка не найдена!";

    public NoSuchLineException() {
        super(message);
    }

    public NoSuchLineException(Throwable cause) {
        super(message, cause);
    }
}
```

NullNameException.java

```java
package core.exception;

public class NullNameException extends Exception{
    private static final String message = "Error! Name can't be empty";


    public NullNameException() {
        super(message);
    }

```

```
    public NullNameException(Throwable cause) {
        super(message, cause);
    }

}
```

NullTextException.java

```
package core.exception;

public class NullTextException extends RuntimeException{
    private static final String message = "Текст не может быть пустым!";


    public NullTextException() {
        super(message);
    }

    public NullTextException(Throwable cause) {
        super(message, cause);
    }
}
```

event

EventDoItInterface.java

```
package core.event;

public interface EventDoItInterface {
    public void setName(String act);
    public String getName();
    public String toString();
    public boolean equals(Object obj);
    public int hashCode();
}
```

Move.java

```
package core.event;

public class Move implements EventDoItInterface {
    private String name;
    public Move(String name){
        this.name=name;
    }

    public Move(String name, boolean do_it){
        if(!do_it){
            this.name="не "+name;
        }
        else this.name=name;
    }


    @Override
    public void setName(String name) {
        this.name = name;
    }

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

    @Override
    public String toString() {
        return "Move{" +
                "name='" + name + '\'' +
                '}';
    }

    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Move){
            return true;
        }
```

```
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,
                    CreatureThingsTimeInterface cre){
14                  this.p = p;
15                  this.s = s;
16                  this.cre = cre;
17              }
18              public void sexDescribe(){
19                  switch (s){
20                      case HE -> {
21                          p.println("На␣сцене␣появился␣" + name,cre);
22                      }
23                      case SHE -> {
24                          p.println("На␣сцене␣появилась␣" + name,cre);
25                      }
26                      case IT -> {
27                          p.println("На␣сцене␣появилось␣" + name,cre);
28                      }
29                      case THEY -> {
30                          p.println("На␣сцене␣появились␣" + name,cre);
31                      }
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␣строчках␣");
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
```

```
    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Neznayka){
            return true;
        }
        return false;
    }
}
```

Person.java

```
package core.CreatureThingsTime.people;


import core.CreatureThingsTime.Creature;
import core.CreatureThingsTime.CreatureThingsTimeInterface;
import core.CreatureThingsTime.Status;
import core.print.CustomPrintInterface;


public abstract class Person extends Creature implements HumanInterface {
    private Status status = Status.NORMAL;
    protected Person(){}
    public Person(String name){
        super.name = name;
    }

    public void setStatus(Status status){
        this.status = status;
        p.println("Статус␣персонажа␣" + this.getName() + "␣изменён␣на␣" + status.getValue());
    }

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

    @Override
    public CreatureType getCreatureType() {
        return CreatureType.People;
    }

    @Override
    public String toString(){
        return name+"␣"+status;
    }

    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Person){
            return true;
        }
        return false;
    }
    @Override
    public int hashCode(){
        int result = name == null ? 0 : name.hashCode();
        result+=status.hashCode();
        return result;
    }
}
```

Someone.java

```
package core.CreatureThingsTime.people;

public class Someone extends Person{
    public Someone() {
        super("кто");
        super.describe(Sex.HE);
    }

    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Neznayka){
            return true;
```

```
        }
        return false;
    }
}
```

They.java

```
package core.CreatureThingsTime.people;

public class They extends Person{
    public They() {
        super.setName("они");
        super.describe(Sex.THEY);
    }

    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Minoga){
            return true;
        }
        return false;
    }
}
```

CreatureThingsTime/dog

Dog.java

```
package core.CreatureThingsTime.dog;

import core.CreatureThingsTime.Creature;

public abstract class Dog extends Creature {
    @Override
    public CreatureType getCreatureType() {
        return CreatureType.Dog;
    }
}
```

Mimi.java

```
package core.CreatureThingsTime.dog;

public class Mimi extends Dog{
    public Mimi() {
        super.setName("Мими");
        super.describe(Sex.SHE);
    }


    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Mimi){
            return true;
        }
        return false;
    }
}
```

Roland.java

```
package core.CreatureThingsTime.dog;
public class Roland extends Dog {
    public Roland() {
        super.setName("Роланд");
        super.describe(Sex.HE);
    }


    @Override
    public boolean equals(Object obj){
        if (this == obj) return true;
        if(obj instanceof Roland){
            return true;
        }
        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 неимущими бедняками и смеялись над ними.
19. В такие вечера на попеченье 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 хлебца 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.