

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



Вариант №311678876767
Лабораторная работа №3
по дисциплине
Программирование

Выполнил Студент группы number
Student's name
Преподаватель:
Teacher's name

г. Санкт-Петербург
2021г.

Содержание

| | | |
|---|-------------------------------------|----|
| 1 | Текст задания | 3 |
| 2 | Диаграмма классов объектной модели. | 4 |
| 3 | Исходный код программы | 5 |
| 4 | Результат выполнения: | 14 |
| 5 | Вывод | 15 |

1 Текст задания

Программа должна удовлетворять следующим требованиям:

- Доработанная модель должна соответствовать принципам SOLID.
- Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
- В разработанных классах должны быть переопределены методы `equals()`, `toString()` и `hashCode()`.
- Программа должна содержать как минимум один перечисляемый тип (`enum`).

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 своем 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 все это голодному Козлику. В первый же день он обратился к госпоже Миноге с просьбой заплатить ему жалованье хотя бы за недельку вперед, так как ему нужно помочь больному приятелю, который находился в дрянингской ночлежке. Госпожа Минога сказала, что теперь он живет 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.

2 Диаграмма классов объектной модели.



3 Исходный код программы

Story.java

```
1 package core;
2
3 import core.event.*;
4 import core.people.HumanInterface;
5 import core.people.Kozlik;
6 import core.people.Minoga;
7 import core.people.Neznayka;
8
9 public class Story {
10
11     public static void main(String[] args) {
12         HumanInterface nez = new Neznayka();
13         HumanInterface koz = new Kozlik();
14         koz.setStatus(Status.SICK);
15         nez.run(new ForgetAbout(koz), false);
16         nez.run(new Visit(koz), false);
17         nez.run(new BeAbleToDo(), true);
18         nez.run(new HaveLunchWith(), true);
19         nez.run(new Eat(), false);
20         nez.run(new Hide(koz), true);
21         koz.setStatus(Status.HUNGRY);
22         nez.run(new Bring(koz), true);
23         HumanInterface mi = new Minoga();
24         nez.run(new Requested(mi), true);
25         nez.run(new Pay(), true);
26         nez.run(new NeedHelp(koz), true);
27         koz.run(new Stay(), true);
28         mi.run(new Say(), true);
29         nez.run(new Living(), true);
30         nez.run(new NeedToBeFriendWith(koz), false);
31         koz.run(new Have(), false);
32         koz.run(new Resides(), true);
33
34     }
35 }
36 }
```

Status.java

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

people

HumanInterface.java

```
1 package core.people;
2
3 import core.Status;
4 import core.event.EventDoItInterface;
5
6 public interface HumanInterface {
7     public void setName(String name);
8
9     public String getName();
10
11     public void describe();
12
13     public void setStatus(Status status);
14
15     public Status getStatus();
16 }
```

```

17     public void run(EventDoItInterface act, boolean i);
18 }

```

Kozlik.java

```

1  package core.people;
2
3  public class Kozlik extends Person {
4      public Kozlik() {
5          super.setName("Козлик");
6          this.describe();
7      }
8
9      public void describe() {
10         System.out.println("На сцене появился Козлик");
11     }
12     @Override
13     public boolean equals(Object obj){
14         if (this == obj) return true;
15         if(obj instanceof Kozlik){
16             return true;
17         }
18         return false;
19     }
20 }

```

Minoga.java

```

1  package core.people;
2
3  public class Minoga extends Person {
4      public Minoga() {
5          super.setName("Госпожа Минога");
6          this.describe();
7      }
8
9      public void describe() {
10         System.out.println("На сцене появилась госпожа Минога");
11     }
12
13     @Override
14     public boolean equals(Object obj){
15         if (this == obj) return true;
16         if(obj instanceof Minoga){
17             return true;
18         }
19         return false;
20     }
21 }

```

Neznayka.java

```

1  package core.people;
2
3  public class Neznayka extends Person {
4      public Neznayka() {
5          super("Незнайка");
6          this.describe();
7      }
8
9      public void describe() {
10         System.out.println("На сцене появился Незнайка");
11     }
12
13     @Override
14     public boolean equals(Object obj){
15         if (this == obj) return true;
16         if(obj instanceof Neznayka){
17             return true;
18         }
19         return false;
20     }
21 }

```

Person.java

```

1  package core.people;
2

```

```

3
4 import core.Status;
5 import core.event.ActionType;
6 import core.event.EventDoItInterface;
7 import core.event.Move;
8
9
10 public abstract class Person implements HumanInterface {
11     private String name;
12     private Status status = Status.NORMAL;
13     protected Person(){}
14     public Person(String name){
15         this.name = name;
16     }
17
18     public void setName(String name){
19         if (!(name.equals(""))){
20             this.name = name;
21         }
22         else{
23             System.out.println("Error! Name can't be empty");
24         }
25     }
26
27     public String getName(){
28         return name;
29     }
30
31     public abstract void describe();//Здесь должно быть описание
32
33     public void setStatus(Status status){
34         this.status = status;
35         System.out.println("Статус персонажа " + this.getName() + " изменён на " + status);
36     }
37
38     public Status getStatus(){
39         return this.status;
40     }
41
42
43     public void run(EventDoItInterface act, boolean i){
44         act.setWho(this);
45         if(i){act.do_it();}
46         else{act.do_not_it();}
47     }
48
49     @Override
50     public String toString(){
51         return name+" "+status;
52     }
53
54     @Override
55     public boolean equals(Object obj){
56         if (this == obj) return true;
57         if(obj instanceof Person){
58             return true;
59         }
60         return false;
61     }
62
63     @Override
64     public int hashCode(){
65         int result = name == null ? 0 : name.hashCode();
66         result+=status.hashCode();
67         return result;
68     }
69 }

```

event

ActionType.java

```

1 package core.event;
2
3 public enum ActionType {
4     VISIT("забежал"),
5     BEABLETODO("удавалось сделать"),
6     HIDE("припрятывал"),
7     FORGETABOUT("забывал"),

```

```

8     HAVELUNCHWITH("обедал"),
9     BRING("относил"),
10    NEEDHELP("нужно_помочь"),
11    NEEDTOBEFRIENDWITH("пристало_водить_компанию"),
12    PAY("заплатить"),
13    EAT("съедал"),
14    REQUESTED("обратился"),
15    STAY("находился"),
16    SAY("сказала"),
17    LIVING("живёт"),
18    HAVE("имеет"),
19    RESIDES("обитает");
20    private String value;
21
22    ActionType(String value) {
23        this.value = value;
24    }
25
26    public String getValue() {
27        return value;
28    }
29 }

```

EventDoItInterface.java

```

1  package core.event;
2
3  import core.people.HumanInterface;
4
5  public interface EventDoItInterface {
6      public void setWho(HumanInterface who);
7
8      public HumanInterface getWho();
9
10     public void setName(ActionType act);
11
12     public String getName();
13
14     public void Do(String act);
15
16     public void do_it();
17
18     public void do_not_it();
19 }

```

Move.java

```

1  package core.event;
2
3  import core.people.HumanInterface;
4
5  public abstract class Move implements EventDoItInterface {
6      private String name;
7      private HumanInterface who;
8
9      public void setWho(HumanInterface who) {
10         this.who = who;
11     }
12
13     public HumanInterface getWho() {
14         return who;
15     }
16
17     public void setName(ActionType act) {
18         this.name = act.getValue();
19     }
20
21     public String getName() {
22         return name;
23     }
24
25     public abstract void Do(String act);
26
27     public void do_it() {
28         Do(name);
29     }
30
31     public void do_not_it() {

```



```

32         Do("не_" + name);
33     }
34
35     @Override
36     public String toString(){
37         return name;
38     }
39
40     @Override
41     public boolean equals(Object obj){
42         if (this == obj) return true;
43         if(obj instanceof Move){
44             return true;
45         }
46         return false;
47     }
48     @Override
49     public int hashCode(){
50         int result = this.getName() == null ? 0 : this.getName().hashCode();
51         return result;
52     }
53 }

```

BeAbleToDo.java

```

1  package core.event;
2
3  public class BeAbleToDo extends Move {
4      public BeAbleToDo() {
5          super.setName(ActionType.BEABLETODO);
6      }
7
8      @Override
9      public void Do(String act) {
10         System.out.printf("Обычно_это_%s_во_время_послеобеденной_прогулки.\n", act);
11     }
12
13     @Override
14     public boolean equals(Object obj){
15         if (this == obj) return true;
16         if(obj instanceof BeAbleToDo){
17             return true;
18         }
19         return false;
20     }
21 }

```

Bring.java

```

1  package core.event;
2
3  import core.people.HumanInterface;
4
5  public class Bring extends Move {
6      private HumanInterface whom;
7
8      public Bring(HumanInterface whom) {
9          super.setName(ActionType.BRING);
10         this.whom = whom;
11     }
12
13     @Override
14     public void Do(String act) {
15         System.out.printf("и_%s_все_это_%s_%s.\n", act, whom.getStatus().getValue(), whom.
16             getName() + "y");
17     }
18     @Override
19     public boolean equals(Object obj){
20         if (this == obj) return true;
21         if(obj instanceof Bring){
22             return true;
23         }
24         return false;
25     }
26 }

```

Eat.java

```

1 package core.event;
2
3 public class Eat extends Move {
4     public Eat() {
5         super.setName(ActionType.EAT);
6     }
7
8     @Override
9     public void Do(String act) {
10         System.out.printf("%s%s свою порцию до конца,\n", super.getWho().getName(), act);
11     }
12     @Override
13     public boolean equals(Object obj){
14         if (this == obj) return true;
15         if(obj instanceof Eat){
16             return true;
17         }
18         return false;
19     }
20 }

```

Forgetabout.java

```

1 package core.event;
2
3 import core.people.HumanInterface;
4
5 public class ForgetAbout extends Move {
6     private HumanInterface whom;
7
8     public ForgetAbout(HumanInterface whom) {
9         super.setName(ActionType.FORGETABOUT);
10        this.whom = whom;
11    }
12
13    @Override
14    public void Do(String act) {
15        System.out.printf("%s никогда %s о своем %s дуре %s.\n", super.getWho().getName(),
16            act, whom.getStatus().getValue(), whom.getName() + "e");
17    }
18    @Override
19    public boolean equals(Object obj){
20        if (this == obj) return true;
21        if(obj instanceof ForgetAbout){
22            return true;
23        }
24        return false;
25    }
26 }

```

HaveLunchWith.java

```

1 package core.event;
2
3 public class HaveLunchWith extends Move {
4     public HaveLunchWith() {
5         super.setName(ActionType.HAVELUNCHWITH);
6     }
7
8     @Override
9     public void Do(String act) {
10        System.out.printf("Всегда, когда %s с собаками,\n", super.getWho().getName(), act);
11    }
12    @Override
13    public boolean equals(Object obj){
14        if (this == obj) return true;
15        if(obj instanceof HaveLunchWith){
16            return true;
17        }
18        return false;
19    }
20 }

```

Hide.java

```

1 package core.event;
2

```

```

3 import core.people.HumanInterface;
4
5 public class Hide extends Move {
6     private HumanInterface whom;
7
8     public Hide(HumanInterface whom) {
9         super.setName(ActionType.HIDE);
10        this.whom = whom;
11    }
12
13    @Override
14    public void Do(String act) {
15        System.out.printf("а_%s_в_карман_то_пирожок," +
16            " _то_котлетку,_то_хлебца_краюшку\n", act);
17    }
18    @Override
19    public boolean equals(Object obj){
20        if (this == obj) return true;
21        if(obj instanceof Hide){
22            return true;
23        }
24        return false;
25    }
26 }

```

Have.java

```

1 package core.event;
2
3 public class Have extends Move {
4     public Have() {
5         super.setName(ActionType.HAVE);
6     }
7
8     @Override
9     public void Do(String act) {
10        System.out.printf("%s_даже_дома_собственного_%s,\n", super.getWho().getName(), act);
11    }
12    @Override
13    public boolean equals(Object obj){
14        if (this == obj) return true;
15        if(obj instanceof Have){
16            return true;
17        }
18        return false;
19    }
20 }

```

Living.java

```

1 package core.event;
2
3 public class Living extends Move {
4     public Living() {
5         super.setName(ActionType.LIVING);
6     }
7
8     @Override
9     public void Do(String act) {
10        System.out.printf("что_теперь_%s_%s_в_богатом_доме,_в_обществе_приличных_собак,\n", super
            .getWho().getName(), act);
11    }
12    @Override
13    public boolean equals(Object obj){
14        if (this == obj) return true;
15        if(obj instanceof Living){
16            return true;
17        }
18        return false;
19    }
20 }

```

NeedHelp.java

```

1 package core.event;
2
3 import core.people.HumanInterface;
4

```

```

5 public class NeedHelp extends Move {
6     private HumanInterface whom;
7
8     public NeedHelp(HumanInterface whom) {
9         super.setName(ActionType.NEEDHELP);
10        this.whom = whom;
11    }
12
13    @Override
14    public void Do(String act) {
15        String who_name = super.getWho().getName().substring(0, super.getWho().getName().
16            length() - 1);
17        who_name += "e";
18
19        System.out.printf("так_как_%s_%s_%s\n", who_name, act, whom.getStatus().getValue
20            (), whom.getName() + "y");
21    }
22    @Override
23    public boolean equals(Object obj){
24        if (this == obj) return true;
25        if(obj instanceof NeedHelp){
26            return true;
27        }
28        return false;
29    }
30 }

```

NeedToBeFriendWith.java

```

1 package core.event;
2
3 import core.people.HumanInterface;
4
5 public class NeedToBeFriendWith extends Move {
6     private HumanInterface whom;
7
8     public NeedToBeFriendWith(HumanInterface whom) {
9         super.setName(ActionType.NEEDTOBEFRIENDWITH);
10        this.whom = whom;
11    }
12
13    @Override
14    public void Do(String act) {
15        String who_name = super.getWho().getName().substring(0, super.getWho().getName().
16            length() - 1);
17        who_name += "e";
18        System.out.printf("и_%s_%s_с_какимто_%s,\n", who_name, act, whom.getName() + "ом");
19    }
20    @Override
21    public boolean equals(Object obj){
22        if (this == obj) return true;
23        if(obj instanceof NeedToBeFriendWith){
24            return true;
25        }
26        return false;
27    }
28 }

```

Pay.java

```

1 package core.event;
2
3 public class Pay extends Move {
4     public Pay() {
5         super.setName(ActionType.PAY);
6     }
7
8     @Override
9     public void Do(String act) {
10        String who_name = super.getWho().getName().substring(0, super.getWho().getName().
11            length() - 1);
12        who_name += "e";
13        System.out.printf("с_просьбой_%s_%s_жалование_хотя_бы_за_недельку_вперед,\n", act,
14            who_name);
15    }
16    public boolean equals(Object obj){
17        if (this == obj) return true;
18        if(obj instanceof Pay){
19            return true;
20        }
21        return false;
22    }
23 }

```

```

17         return true;
18     }
19     return false;
20 }
21 }

```

Requested.java

```

1  package core.event;
2
3  import core.people.HumanInterface;
4
5  public class Requested extends Move {
6      private HumanInterface whom;
7
8      public Requested(HumanInterface whom) {
9          super.setName(ActionType.REQUESTED);
10         this.whom = whom;
11     }
12
13     @Override
14     public void Do(String act) {
15         System.out.printf("В первый же день %s %s %s\n", super.getWho().getName(), act,
16             whom.getName());
17     }
18     @Override
19     public boolean equals(Object obj){
20         if (this == obj) return true;
21         if(obj instanceof Requested){
22             return true;
23         }
24         return false;
25     }
26 }

```

Resides.java

```

1  package core.event;
2
3  public class Resides extends Move {
4      public Resides() {
5          super.setName(ActionType.RESIDES);
6      }
7
8      @Override
9      public void Do(String act) {
10         System.out.printf("а %s в какойто-ночлежке.\n", act);
11     }
12
13     @Override
14     public boolean equals(Object obj){
15         if (this == obj) return true;
16         if(obj instanceof Resides){
17             return true;
18         }
19         return false;
20     }
21 }

```

Say.java

```

1  package core.event;
2
3  public class Say extends Move {
4      public Say() {
5          super.setName(ActionType.SAY);
6      }
7
8      @Override
9      public void Do(String act) {
10         System.out.printf("%s %s\n", super.getWho().getName(), act);
11     }
12     @Override
13     public boolean equals(Object obj){
14         if (this == obj) return true;
15         if(obj instanceof Say){
16             return true;
17         }
18     }
19 }

```

```

18         return false;
19     }
20 }

```

Stay.java

```

1 package core.event;
2
3 public class Stay extends Move {
4     public Stay() {
5         super.setName(ActionType.STAY);
6     }
7
8     @Override
9     public void Do(String act) {
10         System.out.printf("%s%s в дрянингской ночлежке.\n", super.getWho().getName(), act);
11     }
12
13     @Override
14     public boolean equals(Object obj){
15         if (this == obj) return true;
16         if(obj instanceof Stay){
17             return true;
18         }
19         return false;
20     }
21 }

```

Visit.java

```

1 package core.event;
2
3 import core.people.HumanInterface;
4
5 public class Visit extends Move {
6     private HumanInterface whom;
7
8     public Visit(HumanInterface whom) {
9         this.whom = whom;
10        super.setName(ActionType.VISIT);
11    }
12
13    @Override
14    public void Do(String act) {
15        System.out.printf("Не проходило дня, чтоб %s%s, хотя бы на минутку.\n", super.getWho().getName(), act, whom.getName() + "y");
16    }
17
18    @Override
19    public boolean equals(Object obj){
20        if (this == obj) return true;
21        if(obj instanceof Visit){
22            return true;
23        }
24        return false;
25    }
26 }

```

4 Результат выполнения:

На сцене появился Незнайка
 На сцене появился Козлик
 Статус персонажа Козлик изменён на SICK
 Незнайка никогда не забывал о своем больном друге Козлике.
 Не проходило дня, чтоб Незнайка не забежал к Козлику хотя бы на минутку.
 Обычно это удавалось сделать во время послеобеденной прогулки.
 Всегда, когда Незнайка обедал с собаками,
 Незнайка не съедал свою порцию до конца,
 а припрятывал в карман то пирожок, то котлетку, то хлебца краешку
 Статус персонажа Козлик изменён на HUNGRY
 и относил все это голодному Козлику.
 На сцене появилась госпожа Минога
 В первый же день Незнайка обратился к Госпоже Минога
 с просьбой заплатить Незнайке жалование хотя бы за недельку вперед,
 так как Незнайке нужно помочь голодному Козлику
 Козлик находился в дрянингской ночлежке.
 Госпожа Минога сказала,
 что теперь Незнайка живёт в богатом доме, в обществе приличных собак,
 и Незнайке не пристало водить компанию с каким-то Козликом,
 Козлик даже дома собственного не имеет,
 а обитает в какой-то ночлежке.

5 Вывод

В процессе выполнения этой лабораторной работы я познакомился с принципами SOLID. Познакомился с интерфейсами и абстрактными классами в языке Java.