Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники

ИТМО

Вариант №5027
Лабораторная работа №3
по дисциплине
Программирование

Выполнил Студент группы P3112
Кобелев Роман Павлович
Преподаватель:
Гаврилов Антон Валерьевич.

г. Санкт-Петербург
2022г.

1 Задание

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

Продолжив свои наблюдения, давилонские астрономы убедились, что завладевший их вниманием космический предмет постепенно приобрел скорость, достаточную для того, чтоб со временем выйти из сферы земного притяжения. Рассчитав траекторию, то есть линию полета этого перемещающегося в межпланетном пространстве тела, астрономы убедились, что оно направляется к Луне. Об этом немедленно сообщили господину Спрутсу. Господин Спрутс отдал распоряжение продолжать астрономические наблюдения, после чего позвонил по телефону главному полицейскому комиссару Ржиглю и сказал, что ожидается прибытие космического корабля с коротышками на борту, с которыми необходимо как можно скорей разделаться, поскольку они намерены сеять повсюду гигантские семена и подстрекать бедняков к неповиновению богачам. Главный полицейский комиссар Ржигль сказал, что все необходимые меры будут приняты, но просил сообщить о времени ожидаемого прибытия космического корабля на Луну, о месте предполагаемой высадки космонавтов и об их примерном количестве.

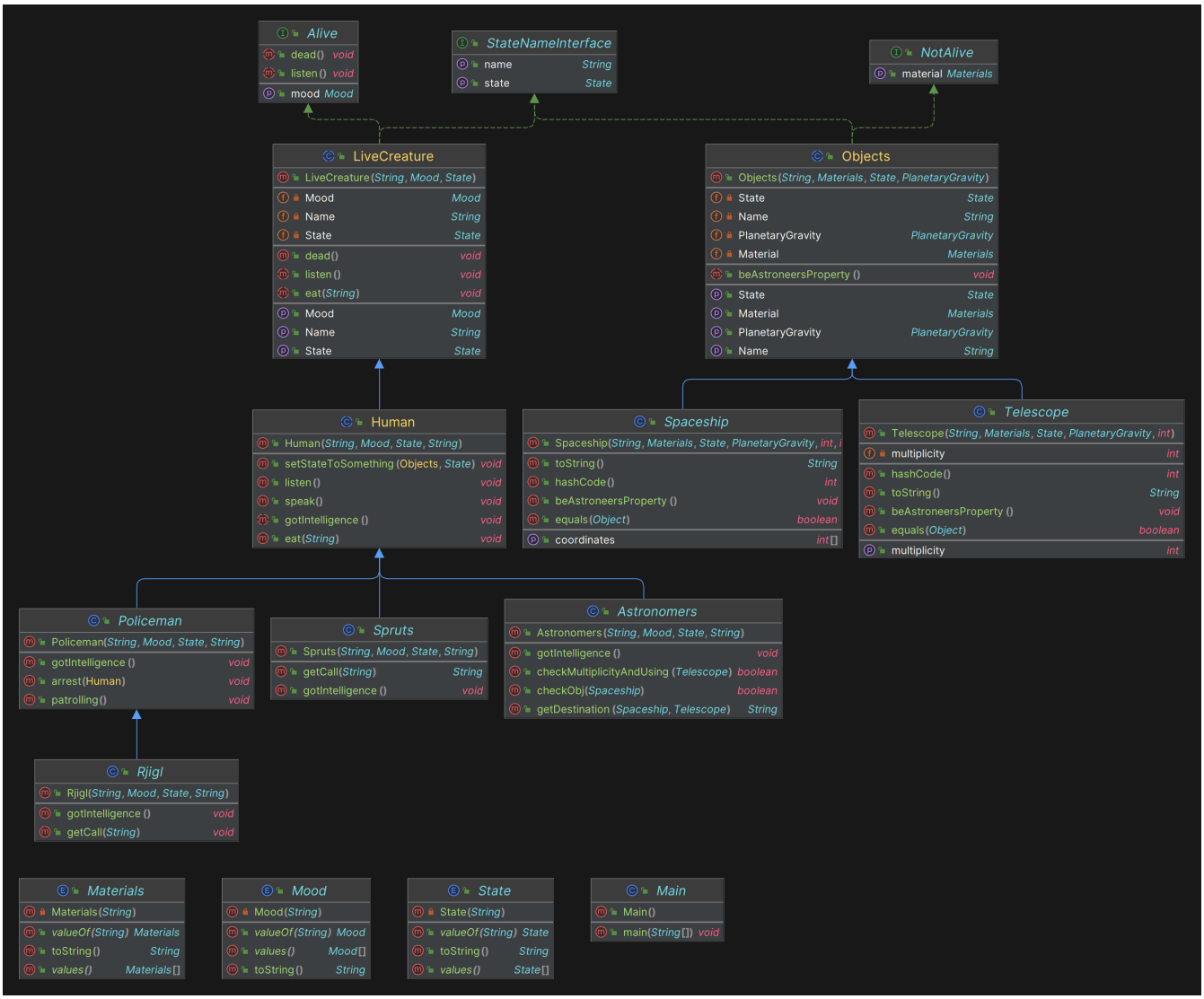
Программа должна удовлетворять следующим требованиям:

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

Порядок выполнения работы:

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

2 Диаграмма классов объектной модели



3 Исходный код программы

Alive.java

```java
package Interfaces;

import Live.Mood;

public interface Alive {
   void dead();
   Mood getMood();
   void setMood(Mood newMood);
   void listen();

}
```

NotAlive.java

```java
package Interfaces;

import Objects.Materials;

public interface NotAlive {
   Materials getMaterial();
}
```

StateNameInterface.java

```java
package Interfaces;

import Others.State;

public interface StateNameInterface {
   String getName();
   State getState();
   void setState(State newState);
}
```

Astronomers.java

```java
package Live;
import Others.State;
import Objects.*;

public class Astronomers extends Human {
   public Astronomers(String name, Mood mood, State state, String nativeLanguage){
      super(name, mood, state, nativeLanguage);
   }

   @Override
   public void gotIntelligence() {
      System.out.println(getName()+"имееет интеллект");
   }

   public boolean checkObj(Spaceship spc){
      return (spc.getState()==State.FLY);

```

```
18      }
19      public boolean checkMultiplicityAndUsing(Telescope tls){
20          return (tls.getState()==State.USE && tls.getMultiplicity() ⩾8);
21      }
22
23      public String getDestination(Spaceship spc, Telescope tls){
24          if (checkObj(spc) && checkMultiplicityAndUsing(tls)){
25              int[] a = spc.getCoordinates();
26              int Velocity = (a[1] - a[0])/(a[2]*60);
27
28              if (Velocity >1600){
29                  setMood(Mood.PANIC);
30                  setState(State.REPORT);
31                  return "MOON";
32              }
33              else
34              {
35                  setMood(Mood.CALM);
36                  return "Something else";
37              }
38          }
39          else{
40              if (!checkMultiplicityAndUsing(tls)){
41                  System.out.println(getName() + ": Возьмидругойтелескоп!");
42              }
43              else{
44                  if (!checkObj(spc)){
45                      System.out.println(getName() + ": Тамничегонет,
                            ↪ чтотыхочешьпросмотреть?");
46                  }
47              }
48
49              return "";
50          }
51
52      }
53
54
55
56
57  }
```

Human.java

```
1   package Live;
2
3   import Objects.*;
4   import Others.State;
5
6   import java.util.concurrent.TimeUnit;
7
8   public abstract class Human extends LiveCreature {
9       private final String nativeLanguage;
10      private final String Name;
11      private State state;
12      public Human(String name, Mood mood, State state, String language){
13          super(name, mood, state);
```

```
14        this.Name = name;
15        this.nativeLanguage = language;
16     }
17     @Override
18     public void listen(){
19        setState(State.LISTEN);
20        try {
21           TimeUnit.SECONDS.sleep(1);
22        } catch (InterruptedException ie) {
23           Thread.currentThread().interrupt();
24        }
25     }
26     @Override
27     public void eat(String nameOfMeal){
28        setState(State.EAT);
29        System.out.println(getName() + ": Сейчасяем"+nameOfMeal);
30
31     }
32     public void speak(){
33        setState(State.TALK);
34        System.out.println(this.Name+" speaking "+this.nativeLanguage);
35        setState(State.STAND);
36     }
37     public void setStateToSomething(Objects object, State newState){
38        object.setState(newState);
39     }
40     public abstract void gotIntelligence();
41
42
43
44  }
```

LiveCreature.java

```
1  package Live;
2
3  import Others.State;
4  import Interfaces.Alive;
5  import Interfaces.StateNameInterface;
6
7  import java.util.concurrent.TimeUnit;
8
9
10 public abstract class LiveCreature implements Alive, StateNameInterface{
11    private final String Name;
12    private Mood Mood;
13    private State State;
14    public LiveCreature(String name, Mood mood, State state){
15       this.Name = name;
16       this.Mood = mood;
17       this.State = state;
18    }
19
20    public final State getState(){
21       return this.State;
22    }
23    public final Mood getMood(){
```

```java
        return this.Mood;
    }
    public final String getName(){
        return this.Name;
    }

    public abstract void eat(String nameOfMeal);
    public abstract void listen();
    public void setState(State newState){
        if (this.getState() == newState){
            System.out.println(this.getName()+" is already "+ this.getState().toString
                ↪ ());
        }
        else{
            this.State = newState;
            System.out.println(this.getName()+" is "+ this.getState().toString()+"
                ↪ right now");
        }
    }
    public final void setMood(Mood newMood){
        if (this.getMood() == newMood){
            System.out.println(this.getName()+" is already "+ this.getMood().toString
                ↪ ());
        }
        else{
            this.Mood = newMood;
            System.out.println(this.getName()+" is "+ this.getMood().toString()+"
                ↪ right now");
        }
    }

    public void dead(){
        this.State = State.DEAD;
    }
}
```

Mood.java

```java
package Live;

public enum Mood {
    HAPPY("happy"),
    SAD("sad"),
    ANGRY("angry"),
    PANIC("panic"),
    CALM("calm"),
    CONCENTRATED("concentrated");

    private final String Status;
    Mood(String status) {
        Status = status;
    }
    @Override
    public final String toString() {
        return this.Status;
    }
}
```

Policeman.java

```java
package Live;

import Others.State;
import java.util.concurrent.TimeUnit;

public class Policeman extends Human{
   public Policeman(String name, Mood mood, State state, String nativeLanguage){
      super(name, mood, state, nativeLanguage);
   }
   @Override
   public void gotIntelligence() {
      System.out.println(getName()+"имееет интеллект");
   }
   public void arrest(Human human){
      human.setState(State.AREST);
      System.out.println(getName() + ": Япоймал"+human.getName()+ "!");
   }
   public void patrolling(){
      setState(State.WALK);
      try {
         TimeUnit.SECONDS.sleep(1);
      } catch (InterruptedException ie) {
         Thread.currentThread().interrupt();
      }
      setState(State.STAND);
   }

}
```

Rjigl.java

```java
package Live;
import Others.State;



public class Rjigl extends Policeman{
   public Rjigl(String name, Mood mood, State state, String nativeLanguage) {
      super(name, mood, state, nativeLanguage);
   }
   @Override
   public void gotIntelligence() {
      System.out.println(getName()+"имееет интеллект");
   }
   public void getCall(String themeOfCall){
      if (themeOfCall == "SPRUTS"){
         listen();
         setState(State.TALK);
         System.out.println(getName() + ": Дайтевсюинформациюобэтомобъекте");
      }
      else{
         System.out.println(getName() + ": Ничегонепроизошло!");
      }
   }
}
```

Spruts.java

```java
package Live;

import Others.State;

public class Spruts extends Human {
   public Spruts(String name, Mood mood, State state, String nativeLanguage){
      super(name, mood, state, nativeLanguage);
   }
   @Override
   public void gotIntelligence() {
      System.out.println(getName()+"не имееетинтеллект");
   }
   public String getCall(String themeOfCall){
      if (themeOfCall == "MOON"){
         listen();
         setMood(Mood.ANGRY);
         setState(State.TALK);
         System.out.println(getName() +": Онихотятнамтогосамогосделать,
            ↪ АЛООПОЛИСМЕНСДЕЛАЙЧТОНИБУДЬ-!");
         return "SPRUTS";

      }
      else{
         System.out.println(getName()+ ": Ничегонепроизошла, аденьг,
            ↪ какивластисталотолькобольше!");
         return "Не надозвноитькомиссару, наэтонетпричин";
      }
   }

}
```

Materials.java

```java
package Objects;

public enum Materials {
   METAL("metal"),
   PLASTIC("plastic"),
   SPACESTONE("spacestone"),
   GLASS("glass");


   private final String name;
   Materials(String name) {
      this.name = name;
   }
   @Override
   public String toString() {
      return this.name;
   }
}
```

Objects.java

```java
package Objects;

```

```java
import Others.State;
import Interfaces.NotAlive;
import Interfaces.StateNameInterface;

public abstract class Objects implements NotAlive, StateNameInterface {
   private final String Name;
   private final Materials Material;
   private State State;
   private PlanetaryGravity PlanetaryGravity;
   public Objects(String name, Materials material, State state, PlanetaryGravity planetaryGravity){
      this.Name = name;
      this.Material = material;
      this.State = state;
      this.PlanetaryGravity = planetaryGravity;
   }

   public abstract void beAstroneersProperty();
   public final State getState(){
      return this.State;
   }
   public final String getName(){
      return this.Name;
   }
   public final Materials getMaterial() {
      return this.Material;
   }
   public PlanetaryGravity getPlanetaryGravity(){
      return this.PlanetaryGravity;
   }
   public void setState(State newState){
      if (this.getState() == newState){
         System.out.println(this.getName()+" is already "+ this.getState().toString());
      }
      else{
         this.State = newState;
         System.out.println(this.getName()+" is "+ this.getState().toString()+" right now");
      }
   }
   public void setPlanetaryGravity(PlanetaryGravity newPlanetaryGravity){
      if (this.getPlanetaryGravity() == newPlanetaryGravity){
         System.out.println(this.getName()+" is already "+ this.getPlanetaryGravity().toString()+ " gravity");
      }
      else{
         this.PlanetaryGravity = newPlanetaryGravity;
         System.out.println(this.getName()+" is in "+ this.getPlanetaryGravity().toString()+" gravity right now");
      }
   }
}
```

Spaceship.java

```java
package Objects;
```

```java

import Others.State;

public class Spaceship extends Objects{
    private final int x_now;
    private final int x_SomeminAgo;
    private final int Minute;
    public Spaceship(String name, Materials material, State state, PlanetaryGravity
        ↪ planetaryGravity, int minute, int x_someminAgo, int x_now){
        super(name, material, state, planetaryGravity);
        this.x_now = x_now;
        this.x_SomeminAgo = x_someminAgo;
        this.Minute = minute;
    }
    @Override
    public void beAstroneersProperty(){
        System.out.println(getName()+" неявляетсясобственностьюлунныхастрономов");
    }
    public int[] getCoordinates(){
        int[] coordinates = {this.x_SomeminAgo, this.x_now, this.Minute};
        return coordinates;
    }

    @Override
    public String toString(){
        return "Это "+getName()+", сделаниз"+getMaterial().toString();
    }
    @Override
    public int hashCode() {
        return this.getCoordinates().hashCode() + getMaterial().hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        if (this.hashCode() == obj.hashCode()) return true;
        if (this.hashCode() ≠ obj.hashCode()) return false;
        return this.toString() == obj.toString();
    }
}
```

Telescope.java

```java
package Objects;

import Others.State;

public class Telescope extends Objects{
    private final int multiplicity;
    public Telescope(String name, Materials material, State state, PlanetaryGravity
        ↪ planetaryGravity, int Multiplicity){
        super(name, material, state, planetaryGravity);
        this.multiplicity = Multiplicity;
    }
    @Override
    public void beAstroneersProperty(){
        System.out.println(getName()+" являетсясобственностьюлунныхастрономов");
    }
```

```
15      public final int getMultiplicity(){
16          return this.multiplicity;
17      }
18
19      @Override
20      public String toString(){
21          return "Этот "+getName()+" сделаниз"+getMaterial().toString()+".
                ↪ Оннужендлянаблюдениязаобъектамивкосмосе.\n"+
22                  "Он имееткратность" + getMultiplicity() + ".";
23      }
24
25      @Override
26      public int hashCode(){
27          return this.getMaterial().hashCode() + this.getName().hashCode() + this.
                ↪ getState().hashCode();
28      }
29      @Override
30      public boolean equals(Object obj) {
31          if (this.hashCode() == obj.hashCode()) return true;
32          if (this.hashCode() ≠ obj.hashCode()) return false;
33          return (this.toString() == obj.toString());
34      }
35  }
```

State.java

```
1   package Others;
2
3   public enum State {
4       STAND("standing"),
5       DEAD("dead"),
6       LISTEN("listening"),
7       REPORT("reporting"),
8       FLY("flying"),
9       SEEK("seeking"),
10      TALK("talking"),
11      WALK("walking"),
12      OBSERVE("observing"),
13      SIT("sitting"),
14      AREST("arrested"),
15      USE("in use"),
16      EAT("eating");
17      private final String Status;
18      State(String status) {
19          Status = status;
20      }
21      @Override
22      public final String toString() {
23          return this.Status;
24      }
25  }
```

Main.java

```
1   import Live.*;
2   import Objects.*;
3   import Others.State;
```

```
4
5  public class Main {
6      public static void main(String[] args) {
7          Telescope telescope = new Telescope("Телескоп", Materials.PLASTIC, State.
               ↪ STAND, PlanetaryGravity.MOON, 10);
8          Astronomers astronomers = new Astronomers("Астрономы", Mood.CONCENTRATED,
               ↪ State.OBSERVE, "moon language");
9          Spaceship spaceship = new Spaceship("кораблик", Materials.METAL, State.FLY,
               ↪ PlanetaryGravity.EARTH, 10, 1, 1000000);
10         Spruts spruts = new Spruts("Спрутс", Mood.HAPPY, State.SIT, "moon language");
11         Rjigl rjigl = new Rjigl("Комиссар Ржигль", Mood.CALM, State.WALK, "moon
               ↪ language");
12
13         astronomers.setStateToSomething(telescope, State.USE);
14         String checkObjectInSpace = astronomers.getDestination(spaceship, telescope);
15         String checkCallToSpruts = spruts.getCall(checkObjectInSpace);
16         rjigl.getCall(checkCallToSpruts);
17         }
18
19     }
```

4 Результат работы программы

Телескоп is in use right now
Астрономы is panic right now
Астрономы is reporting right now
Спрутс is listening right now
Спрутс is angry right now
Спрутс is talking right now
Спрутс: Они хотят нам того самого сделать, АЛОО ПОЛИСМЕН СДЕЛАЙ ЧТО-НИБУДЬ!
Комиссар Ржигль is listening right now
Комиссар Ржигль is talking right now
Комиссар Ржигль: Дайте всю информацию об этом объекте

5 Вывод

Я познакомился с принципами ООП SOLID и STUPID, модификатором abstract, работой с Interface и Enum. Также реализовал наследственность классов.