Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа №4

Вариант №32647.3

Группа: P3132

Выполнил: Цю Тяньшэн

Проверил:
Горбунов Михаил Витальевич

Санкт-Петербург
2021г

Содержание

Постановка задания

По описанию предметной области построить объектную модель:

Такое ускорение движения могло быть объяснено притяжением какой-нибудь другой крупной планеты, но, поскольку вблизи Земли никакой другой планеты не было, оставалось предположить, что обнаруженное тело приобретало ускорение под влиянием какой-то внутренней, то есть находящейся в нем самом, силы. Источником такой силы мог быть работающий реактивный двигатель, и если это так, то обнаруженное космическое тело было не что иное, как космическая ракета.

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.

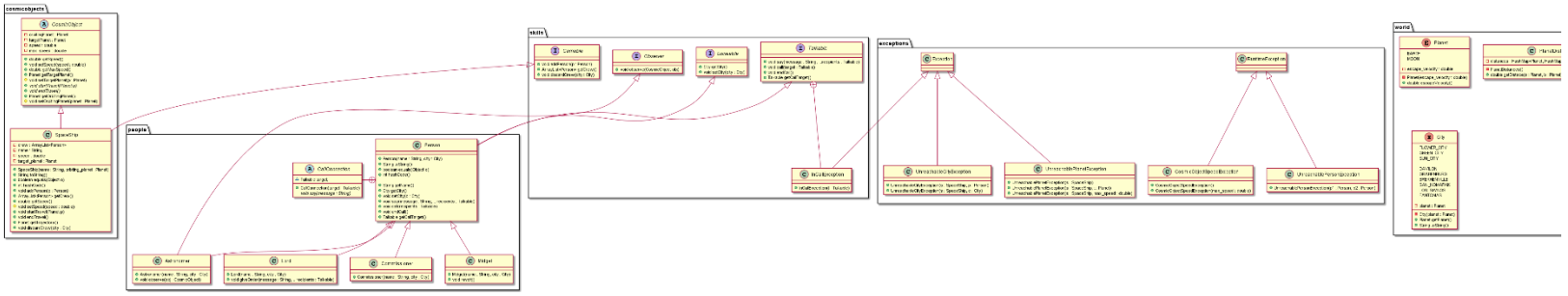
- Все эти сведения необходимы, - сказал он, - чтобы как следует подготовиться к встрече космических гостей и ударить по ним так, чтоб они не успели опомниться.

- Я распоряжусь, чтобы все требуемые сведения были своевременно сообщены вам, - ответил господин Спрутс.

Программа должна удовлетворять следующим требованиям:

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

Диаграмма классов объектной модели



Исходный код программы

exceptions.CosmicObjectSpeedException

```
package exceptions;

public class CosmicObjectSpeedException extends RuntimeException {
    public CosmicObjectSpeedException() {
        super("Cosmic objects cannot have negative speed");
    }

    public CosmicObjectSpeedException(double max_speed) {
        super("Cosmic object is trying to accelerate over maximum speed of " +
max_speed);
    }
}
```

exceptions.UnreachableCityException

```
package exceptions;

import cosmicobjects.SpaceShip;
import people.Person;
import world.City;

public class UnreachableCityException extends Exception {
    public UnreachableCityException(SpaceShip s, Person p) {
        super(p + " can't board " + s + " because they are on different
planets.");
    }

    public UnreachableCityException(SpaceShip s, City c) {
        super(s + " can't discard its crew to " + c + " because they are on
different planets.");
    }
}
```

exceptions.UnreachablePlanetException

```
package exceptions;

import cosmicobjects.SpaceShip;
import world.Planet;

public class UnreachablePlanetException extends Exception {
    public UnreachablePlanetException(SpaceShip s) {
        super(s + " is already orbiting the target planet.");
    }

    public UnreachablePlanetException(SpaceShip s, Planet t) {
        super(t + " is too far away for " + s + " to reach.");
    }

    public UnreachablePlanetException(SpaceShip s, double max_speed) {
        super(s + "can't escape " + s.getOrbitingPlanet() +
```

```
                "because its maximum speed " + max_speed +
                "is less than the required escape velocity of " +
s.getOrbitingPlanet().escapeVelocity());
    }
}
```

exceptions.UnreachablePersonException

```
package exceptions;

import people.Person;

public class UnreachablePersonException extends RuntimeException {
    public UnreachablePersonException(Person p1, Person p2) {
        super(p1 + "can't directly reach " + p2 + " since they are in different
cities.");
    }
}
```

people.Person

```
package people;

import exceptions.UnreachablePersonException;
import skills.Locatable;
import skills.Talkable;
import world.City;

public class Person implements Talkable, Locatable {
    private String name;
    private City city;
    private CallConnection conn;

    private abstract class CallConnection {
        Talkable target;

        public CallConnection(Talkable target) {
            this.target = target;
        }

        abstract void say(String message);
    }

    public Person(String name, City city) {
        this.name = name;
        this.city = city;
    }

    public String toString() {
        return this.getClass().getSimpleName() + ' ' + name + " @ " + city;
    }

    public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (o.getClass() != this.getClass()) return false;

        Person p = (Person)o;

        return this.name == p.name &&
               this.city == p.city;
    }

    public int hashCode() {
        return this.getClass().getName().hashCode() + this.name.hashCode() +
this.city.hashCode();
    }

    public String getName() {
        return name;
    }

    public void setCity(City city) {
        this.city = city;
        System.out.println(this + " is now located at " + city);
        System.out.println();
    }

    public City getCity() {
        return city;
    }

    public void say(String message, Talkable ...recipients) {
        for (Talkable p : recipients) {
            // Сначало проверяем на доступность, т.к. может существовать существо
            // которое находится везде и можно с ним везде разговаривать.

            if (p == getCallTarget()) conn.say(message);
            else if (p instanceof Locatable && ((Locatable)p).getCity() !=
this.getCity())
                throw new UnreachablePersonException(this, (Person)p);
            else System.out.println(this + " says to " + p + " : " + message);
        }

        System.out.println();
    }

    public void call(Talkable target) throws Talkable.InCallException {
        if (this.getCallTarget() != null) throw new
Talkable.InCallException(this);
        if (target.getCallTarget() != null && target.getCallTarget() != this)
throw new Talkable.InCallException(target);

        conn = new CallConnection(target) {
            void say(String message) {
```

```
                System.out.println(Person.this + " says to " + target + " over the
phone: " + message);
            }
        };

        if (target.getCallTarget() == null) {
            System.out.println(this + " establishes a phone connection with " +
target + '\n');
            target.call(this);
        }
    }

    public void endCall() {
        if (getCallTarget() != null) {
            CallConnection temp = conn;
            conn = null;

            if (temp.target.getCallTarget() != null) {
                System.out.println(this + " closes a phone connection with " +
temp.target);
                temp.target.endCall();
            }
        }
    }

    public Talkable getCallTarget() {
        if (conn == null) return null;
        else return conn.target;
    }
}
```

people.Midget

```
package people;
import world.City;

public class Midget extends Person {
    public Midget(String name, City city) {
        super(name, city);
    }

    public void revolt() {
        System.out.println(this + " is now revolting in " + this.getCity() +
'\n');
    }
}
```

people.Lord

```
package people;
import skills.*;
import world.City;

public class Lord extends Person {
```

```
    public Lord(String name, City city) {
        super(name, city);
    }

    public void giveOrder(String message, Talkable ...recipients) {
        for (Talkable p : recipients) {
            if (p instanceof Person)
                System.out.println(this + " gives an order to " + ((Person)p) +
" : " + message);
        }

        System.out.println();
    }
}
```

people.Commissioner

```
package people;
import world.City;

public class Commissioner extends Person {
    public Commissioner(String name, City city) {
        super(name, city);
    }
}
```

people.Astronomer

```
package people;
import cosmicobjects.CosmicObject;
import skills.Observer;
import world.City;
import world.PlanetDistances;

public class Astronomer extends Person implements Observer {
    public Astronomer(String name, City city) {
        super(name, city);
    }

    public void observe(CosmicObject obj) {
        System.out.println(this + " observed a cosmic object in space " + obj);

        if (obj.getTargetPlanet() == null)
            System.out.println("...it is currently standing still!");
        else {
            System.out.println("It is travelling from " + obj.getOrbitingPlanet()
+ " to " + obj.getTargetPlanet() + " at a speed of " + obj.getSpeed() + "m/s");
            System.out.println("It will take " + obj + ' ' +
                               Math.round(PlanetDistances.getDistance(obj.getOrbi
tingPlanet(), obj.getTargetPlanet()) / obj.getSpeed()) +
                               " seconds to reach its destination"
                               );
        }
```

```
        System.out.println();
    }
}
```

cosmicobjects.CosmicObject

```
package cosmicobjects;
import exceptions.CosmicObjectSpeedException;
import exceptions.UnreachablePlanetException;
import world.Planet;

public abstract class CosmicObject {
    private Planet orbitingPlanet;
    private Planet targetPlanet;
    private double speed;
    protected double max_speed = 300000000;

    public abstract String toString();
    public abstract boolean equals(Object o);

    public double getSpeed() {
        return speed;
    }

    protected void setSpeed(double speed) throws CosmicObjectSpeedException {
        if (speed < 0) throw new CosmicObjectSpeedException();
        else if (speed > max_speed) throw new CosmicObjectSpeedException(speed);

        if (this.speed < speed) {
            System.out.println(this + " is increasing its speed by " + (speed -
this.speed) + " m/s");
        } else if (this.speed < speed) {
            System.out.println(this + " is decreasing its speed by " + (this.speed
- speed) + " m/s");
        }

        this.speed = speed;

        System.out.println(this + " is moving at " + getSpeed() + " m/s");
    }

    public double getMaxSpeed() {
        return max_speed;
    }

    public Planet getTargetPlanet() {
        return targetPlanet;
    }

    protected void setTargetPlanet(Planet p) {
        targetPlanet = p;
    }
```

```
    abstract void startTravel(Planet p) throws UnreachablePlanetException;
    abstract void endTravel();

    public Planet getOrbitingPlanet() {
        return orbitingPlanet;
    }

    protected void setOrbitingPlanet(Planet orbitingPlanet) {
        this.orbitingPlanet = orbitingPlanet;
        System.out.println(this + " is now orbiting " + orbitingPlanet);
        System.out.println();
    }
}
```

cosmicobjects.SpaceShip

```
package cosmicobjects;
import java.util.ArrayList;

import people.Person;
import skills.Carriable;
import world.City;
import world.Planet;
import world.PlanetDistances;
import exceptions.*;

public class SpaceShip extends CosmicObject implements Carriable {
    private ArrayList<Person> crew = new ArrayList<Person>();
    private String name;

    public SpaceShip(String name, Planet orbiting_planet) {
        this.name = name;
        setOrbitingPlanet(orbiting_planet);
    }

    public SpaceShip(String name, Planet orbiting_planet, double max_speed) {
        this.name = name;
        setOrbitingPlanet(orbiting_planet);
        this.max_speed = max_speed;
    }

    public String toString() {
        return "SpaceShip \"" + name + "\" with " + crew.size() + " passenger(s) @
" + getOrbitingPlanet().toString();
    }

    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SpaceShip)) return false;

        SpaceShip s = (SpaceShip)o;

        return this.name == s.name &&
```

```
                getSpeed() == s.getSpeed() &&
                this.getTargetPlanet() == s.getTargetPlanet() &&
                this.getOrbitingPlanet() == s.getOrbitingPlanet() &&
                this.crew.equals(s.crew);
    }

    public int hashCode() {
        int hash = (this.name + getSpeed() + this.getTargetPlanet() +
this.getOrbitingPlanet()).hashCode();

        for (Person p : this.crew) hash += p.hashCode();

        return hash;
    }

    @Override
    public void addPerson(Person p) throws UnreachableCityException {
        if (p.getCity().getPlanet() == getOrbitingPlanet()) {
            crew.add(p);
            System.out.println(p + " has boarded " + this + '\n');
        } else {
            throw new UnreachableCityException(this, p);
        }
    }

    public ArrayList<Person> getCrew() {
        return crew;
    }

    public void discardCrew(City city) throws UnreachableCityException {
        if (city.getPlanet() == getOrbitingPlanet()) {
            System.out.println(this + " is discrading its crew to " + city);
            for (Person p : crew) p.setCity(city);
            this.crew.clear();
        } else {
            throw new UnreachableCityException(this, city);
        }
    }

    public void startTravel(Planet p) throws UnreachablePlanetException {
        System.out.println(this + " is trying to move to planet " + p + "...");

        if (getOrbitingPlanet() == p) {
            throw new UnreachablePlanetException(this);
        } else {
            if (Double.isInfinite(PlanetDistances.getDistance(getOrbitingPlanet(),
p)))
                throw new UnreachablePlanetException(this, p);

            try {
                setSpeed(getOrbitingPlanet().escapeVelocity());
```

```
            } catch (CosmicObjectSpeedException e) {
                throw new UnreachablePlanetException(this, this.getMaxSpeed());
            }
            setTargetPlanet(p);
            System.out.println(this + " is traveling towards " + p);
        }

        System.out.println();
    }

    public void endTravel() {
        if (getTargetPlanet() == null) return;

        System.out.println(this + " is approaching " + getTargetPlanet() + "s
orbit!");

        setSpeed(0);

        setOrbitingPlanet(getTargetPlanet());
        setTargetPlanet(null);

        System.out.println();
    }
}
```

skills.Carriable

```
package skills;
import java.util.ArrayList;
import people.Person;
import world.City;
import exceptions.UnreachableCityException;

public interface Carriable {
    void addPerson(Person p) throws UnreachableCityException;
    ArrayList<Person> getCrew();
    void discardCrew(City city) throws UnreachableCityException;
}
```

skills.Locatable

```
package skills;

import world.City;

public interface Locatable {
    City getCity();
    void setCity(City city);
}
```

skills.Observer

```
package skills;
import cosmicobjects.CosmicObject;
```

```
public interface Observer {
    void observe(CosmicObject obj);
}
```

skills.Talkable

```
package skills;

public interface Talkable {
    public class InCallException extends Exception {
        public InCallException(Talkable t) {
            super(t + " is already in a call.");
        }
    }

    String toString();
    void say(String message, Talkable ...recipients);
    void call(Talkable target) throws InCallException;
    void endCall();
    Talkable getCallTarget();
}
```

world.Planet

```
package world;

public enum Planet {
    EARTH(11190.0),
    MOON(2380.0),
    VERY_FAR_PLANET(100.0);

    private final double escape_velocity;

    private Planet(double escape_velocity) {
        this.escape_velocity = escape_velocity;
    }

    public double escapeVelocity() {
        return escape_velocity;
    }
}
```

world.PlanetDistances

```
package world;
import java.util.HashMap;

public final class PlanetDistances {
    private static HashMap<Planet, HashMap<Planet, Double>> distances = new
HashMap<>();

    static {
        for (Planet p : Planet.values())
            distances.put(p, new HashMap<>());
```

```
        //TODO: Put distances in seperate config file
        distances.get(Planet.EARTH).put(Planet.MOON, 384400000.0);
        distances.get(Planet.MOON).put(Planet.EARTH, 384400000.0);
    }

    private PlanetDistances() {}

    public static double getDistance(Planet a, Planet b) {
        if (PlanetDistances.distances.get(a).get(b) == null)
            return (a == b ? 0 : Double.POSITIVE_INFINITY);
        else
            return PlanetDistances.distances.get(a).get(b);
    }
}
```

world.City

```
package world;

public enum City {
    FLOWER_CITY(Planet.EARTH),
    GREEN_CITY(Planet.EARTH),
    SUN_CITY(Planet.EARTH),

    DAVILON(Planet.MOON),
    GRABENBURG(Planet.MOON),
    BREKHENVILLE(Planet.MOON),
    SAN_KOMARAK(Planet.MOON),
    LOS_SWINOS(Planet.MOON),
    FANTOMAS(Planet.MOON);

    private Planet planet;

    private City(Planet planet) {
        this.planet = planet;
    }

    public Planet getPlanet() {
        return this.planet;
    }

    public String toString() {
        return this.name() + " of planet " + this.planet;
    }
}
```

Результат выполнения программы

```
SpaceShip "KOROTYSHKI" with 0 passenger(s) @ EARTH is now orbiting EARTH

Midget Midget 1 @ FLOWER_CITY of planet EARTH has boarded SpaceShip "KOROTYSHKI" with 1 passenger(s) @ EARTH

Midget Midget 2 @ FLOWER_CITY of planet EARTH has boarded SpaceShip "KOROTYSHKI" with 2 passenger(s) @ EARTH

Midget Midget 3 @ FLOWER_CITY of planet EARTH has boarded SpaceShip "KOROTYSHKI" with 3 passenger(s) @ EARTH

Midget Midget 4 @ FLOWER_CITY of planet EARTH has boarded SpaceShip "KOROTYSHKI" with 4 passenger(s) @ EARTH

Midget Midget 5 @ FLOWER_CITY of planet EARTH has boarded SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH

Midget Midget 6 @ DAVILON of planet MOON can't board SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH because they are on different planets.
Astronomer Astronomer 1 @ DAVILON of planet MOON observed a cosmic object in space SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH
...it is currently standing still!

SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is trying to move to planet EARTH...
SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is already orbiting the target planet.
SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is trying to move to planet VERY_FAR_PLANET...
VERY_FAR_PLANET is too far away for SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH to reach.
SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is trying to move to planet MOON...
SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is increasing its speed by 11190.0 m/s
SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is moving at 11190.0 m/s
SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH is traveling towards MOON

Astronomer Astronomer 2 @ DAVILON of planet MOON observed a cosmic object in space SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH
It is travelling from EARTH to MOON at a speed of 11190.0m/s
It will take SpaceShip "KOROTYSHKI" with 5 passenger(s) @ EARTH 34352 seconds to reach its destination

Astronomer Astronomer 2 @ DAVILON of planet MOON says to Lord Spruts @ DAVILON of planet MOON : We have observed a cosmic object moving towards the Moon from Earth

Lord Spruts @ DAVILON of planet MOON gives an order to Astronomer Astronomer 1 @ DAVILON of planet MOON : Continue your observation
Lord Spruts @ DAVILON of planet MOON gives an order to Astronomer Astronomer 2 @ DAVILON of planet MOON : Continue your observation

Lord Spruts @ DAVILON of planet MOON establishes a phone connection with Commissioner Ergigel @ DAVILON of planet MOON

Lord Spruts @ DAVILON of planet MOON says to Commissioner Ergigel @ DAVILON of planet MOON over the phone: A ship full of midgets is arriving soon, we have to deal with it ASAP

Commissioner Ergigel @ DAVILON of planet MOON says to Lord Spruts @ DAVILON of planet MOON over the phone: All necessary measures will be taken

Commissioner Ergigel @ DAVILON of planet MOON says to Lord Spruts @ DAVILON of planet MOON over the phone: We need to know the ETA of the ship, where there are landing and how many of them are there

Commissioner Ergigel @ DAVILON of planet MOON says to Lord Spruts @ DAVILON of planet MOON over the phone: The information is necessary, so we could strike before they have the time to prepare

Lord Spruts @ DAVILON of planet MOON says to Commissioner Ergigel @ DAVILON of planet MOON over the phone: I will work on that, the information will be provided to you in time.

Lord Spruts @ DAVILON of planet MOON closes a phone connection with Commissioner Ergigel @ DAVILON of planet MOON
```

Вывод

В процессе выполнения данной работы были получены навыки о использовании исключении, а также создание анонимных, вложенных и локальных классов в языке Java.