Университет ИТМО

Факультет ПИиКТ

Студент Птицын Максим Евгеньевич

Группа P3130

Преподаватель Блохина Елена Николаевна

Лабораторная работа №4.

Вариант № 20279.

г. Санкт-Петербург

2021 г.

Текст задания:

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

Снусмумрик поставил шляпу на пол между комодом и кухонной дверью.

Муми-тролль и, прежде чем выйти в сад, бросил яичную скорлупу в новую корзину для бумаг, потому что (иногда) он был очень аккуратный муми-тролль.

Гостиная опустела. А в углу, между комодом и дверью на кухню, осталась шляпа Волшебника с яичной скорлупой. И тут сотворилось чудо: яичная скорлупа начала преображаться.

Дело в том, что всякая вещь, если она достаточно долго пролежит в шляпе Волшебника, превращается в нечто совершенно иное — и никогда нельзя знать заранее, во что именно. Муми-папе ужасно повезло, что шляпа ему не подошла: побудь он в ней чуточку подольше — и только покровителю всех троллей и сниффов известно, какая участь его ожидала.

Муми-папа заработал лишь легкую головную боль (которая прошла после обеда). Зато яичные скорлупки, оставшиеся в шляпе, мало-помалу начали менять свой вид. Они сохранили белый цвет, но все росли и росли в размерах и стали мягкими и пухлыми. Немного погодя они целиком заполнили шляпу, а потом из шляпы выпорхнули пять маленьких круглых тучек. Они выплыли на веранду, мягко спустились с крыльца и повисли в воздухе над самой землей. А в шляпе Волшебника стало пусто.

Тучки неподвижно стояли перед ними и словно чего-то ждали.

Фрекен Снорк тихонечко протянула лапу и потрогала тучку, которая была к ней поближе.

Тут все придвинулись ближе и стали ощупывать тучки.

Снусмумрик осторожно толкнул одну из тучек. Она проплыла немного в воздухе и снова застыла на месте.

Муми-тролль только покачал головой в ответ.

фрекен Снорк прижала тучку к земле и погладила ее лапами. В следующее мгновение она уже сидела на тучке и с хихиканьем подскакивала на ней.

Снифф и мигом взобрался на другую тучку

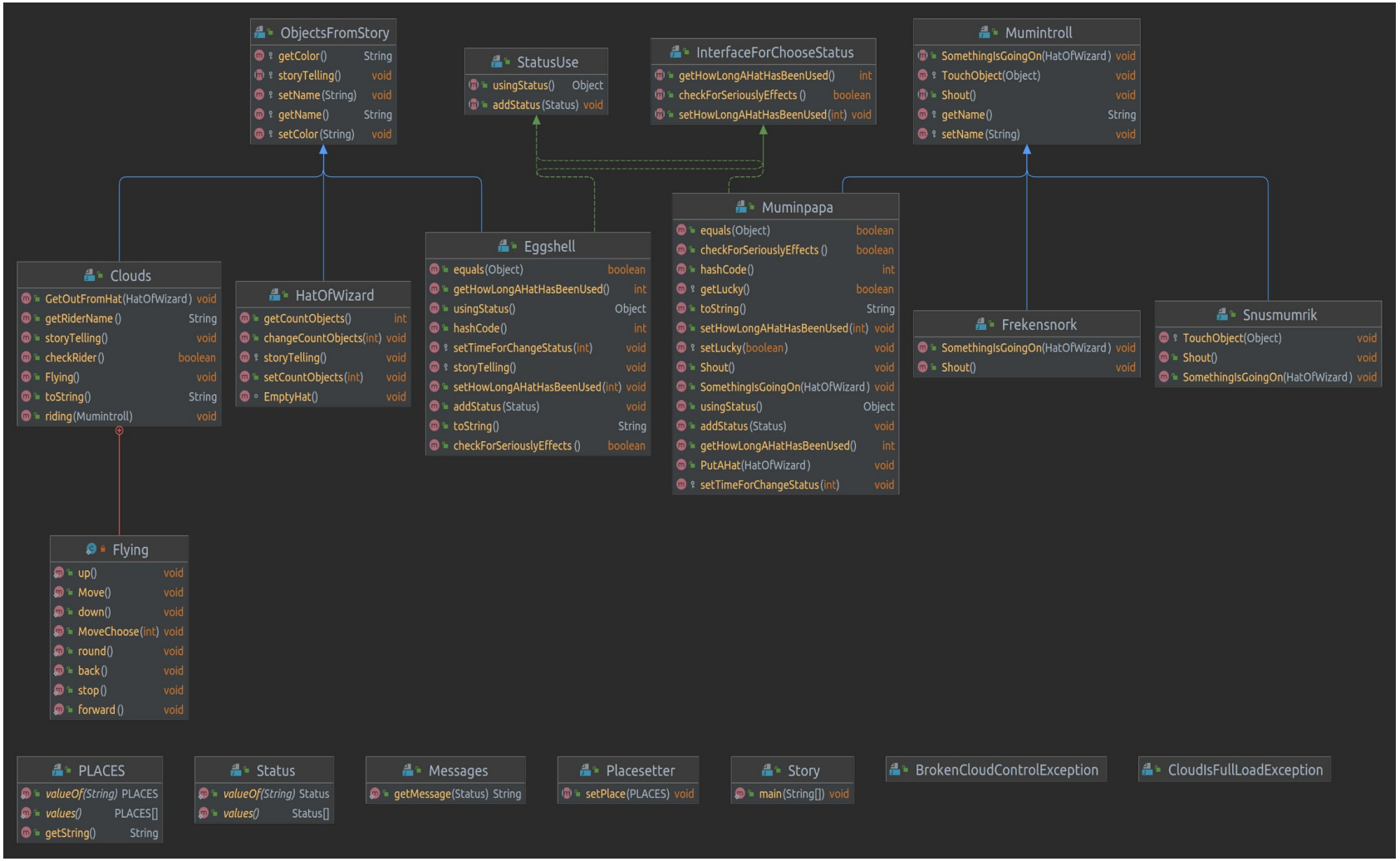
И только он крикнул: «А ну давай!» — как тучка поднялась над землей и описала небольшую изящную дугу.

Муми-тролль и фрекен Снорк катаются на облаках.

Тут уж и все остальные взобрались каждый на свою тучку и закричали: «А ну давай! Гоп!»

Тучки, словно большие послушные кролики, парили над землей. Ими можно было управлять — это открытие сделал Снорк. Легкий нажим одной ногой — поворот. Обеими ногами — полный вперед. Чуть покачаешь тучку — и она набирает высоту.

1. **Программа должна удовлетворять следующим требованиям:**
    a. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
    b. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).
2. **Порядок выполнения работы:**
    a. Доработать объектную модель приложения.
    b. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
    c. Согласовать с преподавателем изменения, внесённые в модель.
    d. Модифицировать программу в соответствии с внесёнными в модель изменениями.


ObjectsFromStory
getColor() String
storyTelling() void
setName(String) void
getName() String
setColor(String) void
StatusUse
usingStatus() Object
addStatus(Status) void
InterfaceForChooseStatus
getHowLongAHatHasBeenUsed() int
checkForSeriouslyEffects() boolean
setHowLongAHatHasBeenUsed(int) void
Mumintroll
SomethingIsGoingOn(HatOfWizard) void
TouchObject(Object) void
Shout() void
getName() String
setName(String) void
Clouds
GetOutFromHat(HatOfWizard) void
getRiderName() String
storyTelling() void
checkRider() boolean
Flying() void
toString() String
riding(Mumintroll) void
HatOfWizard
getCountObjects() int
changeCountObjects(int) void
storyTelling() void
setCountObjects(int) void
EmptyHat() void
Eggshell
equals(Object) boolean
getHowLongAHatHasBeenUsed() int
usingStatus() Object
hashCode() int
setTimeForChangeStatus(int) void
storyTelling() void
setHowLongAHatHasBeenUsed(int) void
addStatus(Status) void
toString() String
checkForSeriouslyEffects() boolean
Muminpapa
equals(Object) boolean
checkForSeriouslyEffects() boolean
hashCode() int
getLucky() boolean
toString() String
setHowLongAHatHasBeenUsed(int) void
setLucky(boolean) void
Shout() void
SomethingIsGoingOn(HatOfWizard) void
usingStatus() Object
addStatus(Status) void
getHowLongAHatHasBeenUsed() int
PutAHat(HatOfWizard) void
setTimeForChangeStatus(int) void
Frekensnork
SomethingIsGoingOn(HatOfWizard) void
Shout() void
Snusmumrik
TouchObject(Object) void
Shout() void
SomethingIsGoingOn(HatOfWizard) void
Flying
up() void
Move() void
down() void
MoveChoose(int) void
round() void
back() void
stop() void
forward() void
PLACES
valueOf(String) PLACES
values() PLACES[]
getString() String
Status
valueOf(String) Status
values() Status[]
Messages
getMessage(Status) String
Placesetter
setPlace(PLACES) void
Story
main(String[]) void
BrokenCloudControlException
CloudIsFullLoadException

Исходный код:

Story.java

```
public class Story {
    public static void main(String[] args) throws Exception {
        HatOfWizard hat = new HatOfWizard();
        Muminpapa papa = new Muminpapa();
        papa.PutAHat(hat);
        Snusmumrik snusmumrik = new Snusmumrik();
        snusmumrik.SomethingIsGoingOn(hat);
        papa.SomethingIsGoingOn(hat);
        Eggshell[] eggshells = new Eggshell[5];
        for (int i = 0; i < eggshells.length; i++) {
            eggshells[i] = new Eggshell();
        }
        papa.usingStatus();
        Clouds[] clouds = new Clouds[5];
        for (int i = 0; i < eggshells.length; i++) {
            clouds[i] = (Clouds) eggshells[i].usingStatus();
        }
        for (int i = 0; i< clouds.length; i++){
            clouds[i].GetOutFromHat(hat);
        }
        hat.EmptyHat();
        Frekensnork frekensnork = new Frekensnork();
        papa.placesetter.setPlace(PLACES.GARDEN);
        snusmumrik.placesetter.setPlace(PLACES.GARDEN);
        frekensnork.TouchObject(clouds[0]);
        snusmumrik.TouchObject(clouds[1]);
        papa.TouchObject(clouds[2]);
        clouds[0].riding(frekensnork);
        clouds[1].riding(snusmumrik);
        clouds[2].riding(papa);
        for (Clouds c: clouds) {
            if (!(c.checkRider())) {
                try {
                    c.Flying();
                } catch (BrokenCloudControlException e) {
                    System.out.println(c.getRiderName() + e.getMessage());
                    break;
                }
            }
        }
    }
}
```

StatusUse.java

```
public interface StatusUse {
    Object usingStatus() throws Exception;
    void addStatus (Status c);
}
```

InterfaceForChooseStatus.java

```
public interface InterfaceForChooseStatus {
    void setHowLongAHatHasBeenUsed(int time);
    int getHowLongAHatHasBeenUsed();
    boolean checkForSeriouslyEffects();
}
```

Placesetter.java

```
@FunctionalInterface
public interface Placesetter {
    void setPlace(PLACES place);
}
```

Mumintroll.java

```
public abstract class Mumintroll{
    private String name;
    private PLACES PLACE;
    protected Mumintroll(){
    }
    protected String getName(){
        return this.name;
    }
    protected void setName(String name){
        this.name=name;
    }
    protected void TouchObject(Object o){
        System.out.println(name + " трогает " + o.toString());
    }
    Placesetter placesetter = (place) {
        PLACE = place;
        System.out.println(name + " перемещается в " + PLACE.getString());
    };
    public abstract void SomethingIsGoingOn(HatOfWizard hat) throws Exception;
    public abstract void Shout();
}
```

Muminpapa.java

```
import java.util.ArrayList;
public class Muminpapa extends Mumintroll implements InterfaceForChooseStatus,
StatusUse{
    private boolean isLucky;
    private int HowLongAHatHasBeenOnHead;
    private int TimeForChangeStatus;
    private final ArrayList<Status> StatusArrayList = new ArrayList<>();
    public Muminpapa(){
        setName("Мумми-папа");
        setLucky(true);
        setTimeForChangeStatus(3000);
    }
    public void PutAHat(HatOfWizard hat) throws Exception {
        System.out.println(getName() + " надевает шляпу Волшебника.");
        hat.changeCountObjects(1);
        if (getLucky()) {
            System.out.print("Шляпа ему не подошла. ");
            System.out.println(getName() + " носит шляпу непродолжительное время.
");
            setHowLongAHatHasBeenUsed(500);
            Thread.sleep(getHowLongAHatHasBeenUsed());
            System.out.println(getName() + " снимает шляпу. ");
            hat.changeCountObjects(-1);
        } else {
            System.out.println("Мумми-папе не повезло, через какое-то время он
превратится во что-то неизвестное.");
            Thread.currentThread().interrupt();
        }
    }
    @Override
    public void SomethingIsGoingOn(HatOfWizard hat) throws Exception {
        System.out.println("Проходит какое-то время, Мумми-папа выбрасывает
скорлупки в шляпу...");
        hat.changeCountObjects(5);
        Thread.sleep(500);
    }
    @Override
    public void Shout() {
        System.out.println(getName() + " кричит:\"Вперёд!\"");
    }
    public void addStatus (Status c){
        this.StatusArrayList.add(c);
    }
    public void setHowLongAHatHasBeenUsed(int time){
        this.HowLongAHatHasBeenOnHead = time;
    }
    public int getHowLongAHatHasBeenUsed(){
        return this.HowLongAHatHasBeenOnHead;
    }
    public boolean checkForSeriouslyEffects() {
        return equals(this);
    }
```

```
    protected void setLucky(boolean luck){
        this.isLucky=luck;
    }
    protected boolean getLucky(){
        return this.isLucky;
    }
    @Override
    public Object usingStatus() {
        if (checkForSeriouslyEffects()) {
            System.out.println("происходят серьёзные изменения...");
            return this;
        } else {
            addStatus(Status.HEADACHE);
            System.out.println(getName() + " ощущает " + this.toString() + " .");
            return this;
        }
    }
    @Override
    public int hashCode() {
        return TimeForChangeStatus-HowLongAHatHasBeenOnHead;
    }
    @Override
    public boolean equals(Object o){
        return o.hashCode()<=0;
    }
    @Override
    public String toString(){
        switch (StatusArrayList.size()){
            case 1:{
                return Messages.getMessage(StatusArrayList.get(0));
            }
            case 2:{
                return Messages.getMessage(StatusArrayList.get(0))+" и
"+Messages.getMessage(StatusArrayList.get(1));
            }
            case 3:{
                return Messages.getMessage(StatusArrayList.get(0))+",
"+Messages.getMessage(StatusArrayList.get(1))+" и
"+Messages.getMessage(StatusArrayList.get(2));
            }
            default:{
                return "здоров";
            }
        }
    }
    protected void setTimeForChangeStatus(int i){
        this.TimeForChangeStatus=i;
    }
}
```

Snusmumrik.java

```
public class Snusmumrik extends Mumintroll{
    public Snusmumrik() {
        setName("Снусмумрик");
        placesetter.setPlace(PLACES.LIVINGROOM);
    }
    @Override
    public void SomethingIsGoingOn(HatOfWizard hat){
        System.out.println("Снусмумрик взял шляпу и поставил её на пол между комодом
и кухонной дверью.");
        hat.placesetter.setPlace(PLACES.LIVINGROOM);
    }
    @Override
    public void Shout() {
        System.out.println(getName() + " кричит:\"А ну давай!\"");
    }
    @Override
    protected void TouchObject(Object o){
        System.out.println(getName() + " толкает " + o.toString());
        System.out.println("Она проплывает небольшое расстояние. ");
    }
}
```

Frekensnork.java

```
public class Frekensnork extends Mumintroll{
    public Frekensnork(){
        setName("Фрекенснорк");
        placesetter.setPlace(PLACES.GARDEN);
    }
    @Override
    public void SomethingIsGoingOn(HatOfWizard hat){
        System.out.println(getName()+" прижимает " + hat.getName()+" к земле и
запрыгивает на неё.");
    }
    @Override
    public void Shout() {
        System.out.println(getName() + " кричит:\"А ну давай! Гоп!\"");
    }
}
```

ObjectsFromStory.java

```
public abstract class ObjectsFromStory{
    private String name;
    private String color;
    private PLACES PLACE;
    protected String getName(){
        return this.name;
    }
    protected void setName(String name){
        this.name = name;
    }
    protected void setColor(String color){
        this.color = color;
    }
    protected String getColor(){
        return this.color;
    }
    Placesetter placesetter = new Placesetter() {
        @Override
        public void setPlace(PLACES place) {
            PLACE = place;
            System.out.println(name + " перемещается в " + PLACE.getString());
        }
    };
    protected abstract void storyTelling() throws Exception;
}
```

HatOfWizard.java

```
public class HatOfWizard extends ObjectsFromStory{
    private int CountObjects;
    public HatOfWizard() {
        setName("Шляпа Волшебника");
        setCountObjects(0);
        storyTelling();
    }
    public void setCountObjects(int countObjects) {
        this.CountObjects = countObjects;
    }
    public int getCountObjects() {
        return CountObjects;
    }
    public void changeCountObjects(int i) {
        this.CountObjects = CountObjects + i;
    }
    void EmptyHat() {
        if (getCountObjects() == 0) {
            System.out.println("Шляпа пуста.");
        } else {
            System.out.println("Шляпа заполнена.");
        }
    }
```

```
    @Override
    protected void storyTelling(){
        System.out.println("Всякая вещь, если она достаточно долго пролежит в шляпе Волшебника, превращается в нечто совершенно иное - и никогда нельзя знать заранее, во что именно.");
    }
}
```

Eggshell.java

```
import java.util.ArrayList;
public class Eggshell extends ObjectsFromStory implements InterfaceForChooseStatus, StatusUse{
    private int HowLongAHatHasBeenOnHead;
    private int TimeForChangeStatus;
    private ArrayList<Status> StatusArrayList = new ArrayList<>();
    public Eggshell() throws Exception {
        setTimeForChangeStatus(5000);
        setName("Яичная скорлупка");
        setColor("Белый");
        storyTelling();
    }
    protected void storyTelling() throws Exception{
        placesetter.setPlace(PLACES.HAT);
        System.out.println(getName() + " лежит в шляпе...");
        Thread.sleep(500);
        setHowLongAHatHasBeenUsed(5000);
    }
    public void setHowLongAHatHasBeenUsed(int time){
        this.HowLongAHatHasBeenOnHead = time;
    }
    public int getHowLongAHatHasBeenUsed(){
        return this.HowLongAHatHasBeenOnHead;
    }
    public boolean checkForSeriouslyEffects() {
        return equals(this);
    }
    protected void setTimeForChangeStatus(int i){
        this.TimeForChangeStatus=i;
    }
    @Override
    public Object usingStatus() throws Exception {

        if (checkForSeriouslyEffects()) {
            addStatus(Status.SOFT);
            addStatus(Status.PLUMP);
            System.out.println(getName() + " становится " + this.toString()+" ...");
            Thread.sleep(500);
            System.out.println(getName() + " увеличивается в размерах и превращается в тучку.");
            Clouds cloud = new Clouds(getColor(), StatusArrayList.get(0), StatusArrayList.get(1));
            return cloud;
        } else {
```

```
            System.out.println("Ничего не происходит...");
            return this;
        }
    }
    @Override
    public int hashCode() {
        return TimeForChangeStatus-HowLongAHatHasBeenOnHead;
    }
    @Override
    public boolean equals(Object o){
        return o.hashCode()<=0;
    }
    public void addStatus (Status c){
        this.StatusArrayList.add(c);
    }
    @Override
    public String toString(){
        switch (StatusArrayList.size()){
            case 1:{
                return Messages.getMessage(StatusArrayList.get(0));
            }
            case 2:{
                return Messages.getMessage(StatusArrayList.get(0))+" и
"+Messages.getMessage(StatusArrayList.get(1));
            }
            case 3:{
                return Messages.getMessage(StatusArrayList.get(0))+",
"+Messages.getMessage(StatusArrayList.get(1))+" и
"+Messages.getMessage(StatusArrayList.get(2));
            }
            default:{
                return "здоров";
            }
        }
    }
}
```

Clouds.java

```
import java.util.ArrayList;
public class Clouds extends ObjectsFromStory{
    private ArrayList<Status> StatusArrayList = new ArrayList<>();
    private Mumintroll rider = null;
    public Clouds(String color, Status s1, Status s2) throws Exception {
        setName("Тучка");
        setColor(color);
        StatusArrayList.add(s1);
        StatusArrayList.add(s2);
        storyTelling();
    }
    @Override
    public void storyTelling() throws Exception{
        System.out.println(getName() + " взлетает из шляпы и мягко парит в
воздухе.");
        Thread.sleep(500);
```

```
    }
    public void GetOutFromHat(HatOfWizard hat) throws Exception{
        System.out.println(getName() + " выплывает на веранду и повисает над самой
землёй.");
        hat.changeCountObjects(-1);
        placesetter.setPlace(PLACES.GARDEN);
        System.out.println(getName() + " ждёт...");
        Thread.sleep(500);
    }
    @Override
    public String toString(){
        return getName();
    }
    public void riding(Mumintroll troll) {
        if (checkRider()){
        rider = troll;
        System.out.println(troll.getName() + " оседлал " + getName());
        } else {
            throw new CloudIsFullLoadException(troll.getName() +" пробует оседлать
"+ getName() + ", но она занята. ");
        }
    }
    public boolean checkRider(){
        return (rider==null);
    }
    public String getRiderName(){
        return rider.getName();
    }
    public void Flying() throws BrokenCloudControlException {
        System.out.println(rider.getName() + " взлетает и пытается управлять тучкой.
");
        for (int i=0; i<5; i++){
            Flying.Move();
            }
        rider.Shout();
        }
    private static class Flying {
        public static void Move() throws BrokenCloudControlException {
            class MoveNumber {
                private int move;
                MoveNumber(){
                    this.move=(int) (Math.random()*5 + 1);
                }
                int getNumber(){
                    return this.move;
                }
            }
            MoveNumber number = new MoveNumber();
            MoveChoose(number.getNumber());
        }
        public static void MoveChoose(int move) throws BrokenCloudControlException {
            switch (move){
            case 1:
```

```
                up();
                break;
            case 2:
                down();
                break;
            case 3:
                forward();
                break;
            case 4:
                back();
                break;
            case 5:
                round();
                break;
            case 6:
                stop();
                break;
            }
        }
        public static void up() throws BrokenCloudControlException {
            if(Math.random()<0.05){
                throw new BrokenCloudControlException(" не справляется с
управлением, падает и разбивается насмерть. ");
            } else {
                System.out.println("Взлетает выше. ");
            }
        }
        public static void down() throws BrokenCloudControlException {
            if(Math.random()<0.05){
                throw new BrokenCloudControlException(" не справляется с
управлением, падает и разбивается насмерть. ");
            } else {
                System.out.println("Опускается ниже. ");
            }
        }
        public static void forward() throws BrokenCloudControlException {
            if(Math.random()<0.05){
                throw new BrokenCloudControlException(" не справляется с
управлением, падает и разбивается насмерть. ");
            } else {
                System.out.println("Ускоряется. ");
            }
        }
        public static void back() throws BrokenCloudControlException {
            if(Math.random()<0.05){
                throw new BrokenCloudControlException(" не справляется с
управлением, падает и разбивается насмерть. ");
            } else {
                System.out.println("Даёт задний ход. ");
            }
        }
        public static void round() throws BrokenCloudControlException {
            if(Math.random()<0.05){
```

```
                throw new BrokenCloudControlException(" не справляется с
управлением, падает и разбивается насмерть. ");
            } else {
                System.out.println("Поворачивает. ");
            }
        }
        public static void stop() throws BrokenCloudControlException {
            if(Math.random()<0.05){
                throw new BrokenCloudControlException(" не справляется с
управлением, падает и разбивается насмерть. ");
            } else {
                System.out.println("Останавливается. ");
            }
        }
    }
}
```

BrokenCloudControlException.java

```
public class BrokenCloudControlException extends Exception{
    public BrokenCloudControlException(String ErrorMessage){
        super(ErrorMessage);
    }
}
```

CloudIsFullLoadException.java

```
public class CloudIsFullLoadException extends RuntimeException{
    public CloudIsFullLoadException (String ErrorMessage) {
        super(ErrorMessage);
    }
}
```

Messages.java

```
public class Messages {
    public Messages() {}
    public static String getMessage(Status c){
        switch (c) {
            case SOFT: {
                return "мягкая";
            }
            case PLUMP: {
                return "пухлая";
            }
            case HEADACHE: {
                return "головную боль";
            }
        }
        return "ничего";
    }
}
```

Status.java

```
public enum Status {
    HEADACHE,
    SOFT,
    PLUMP;
}
```

PLACES.java

```
public enum PLACES {
    LIVINGROOM("Гостиная"),
    GARDEN("Сад"),
    KITCHEN("Кухня"),
    HAT("Шляпа");

    private final String string;

    PLACES(String s) {
        this.string=s;
    }

    public String getString(){
        return string;
    }
}
```

Результат работы программы:

Всякая вещь, если она достаточно долго пролежит в шляпе Волшебника, превращается в нечто совершенно иное - и никогда нельзя знать заранее, во что именно.

Мумми-папа надевает шляпу Волшебника.

Шляпа ему не подошла. Мумми-папа носит шляпу непродолжительное время.

Мумми-папа снимает шляпу.

Снусмумрик перемещается в Гостиная

Снусмумрик взял шляпу и поставил её на пол между комодом и кухонной дверью.

Шляпа Волшебника перемещается в Гостиная

Проходит какое-то время, Мумми-папа выбрасывает скорлупки в шляпу...

Яичная скорлупка перемещается в Шляпа

Яичная скорлупка лежит в шляпе...

Яичная скорлупка перемещается в Шляпа

Яичная скорлупка лежит в шляпе...

Яичная скорлупка перемещается в Шляпа

Яичная скорлупка лежит в шляпе...

Яичная скорлупка перемещается в Шляпа

Яичная скорлупка лежит в шляпе...

Яичная скорлупка перемещается в Шляпа

Яичная скорлупка лежит в шляпе...

Мумми-папа ощущает головную боль .

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Тучка взлетает из шляпы и мягко парит в воздухе.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Тучка взлетает из шляпы и мягко парит в воздухе.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Тучка взлетает из шляпы и мягко парит в воздухе.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Тучка взлетает из шляпы и мягко парит в воздухе.

Яичная скорлупка становится мягкая и пухлая ...

Яичная скорлупка увеличивается в размерах и превращается в тучку.

Тучка взлетает из шляпы и мягко парит в воздухе.

Тучка выплывает на веранду и повисает над самой землёй.

Тучка перемещается в Сад

Тучка ждёт...

Тучка выплывает на веранду и повисает над самой землёй.

Тучка перемещается в Сад

Тучка ждёт...

Тучка выплывает на веранду и повисает над самой землёй.

Тучка перемещается в Сад

Тучка ждёт...

Тучка выплывает на веранду и повисает над самой землёй.

Тучка перемещается в Сад
Тучка ждёт...
Тучка выплывает на веранду и повисает над самой землёй.
Тучка перемещается в Сад
Тучка ждёт...
Шляпа пуста.
Фрекенснорк перемещается в Сад
Мумми-папа перемещается в Сад
Снусмумрик перемещается в Сад
Фрекенснорк трогает Тучка
Снусмумрик толкает Тучка
Она проплывает небольшое расстояние.
Мумми-папа трогает Тучка
Фрекенснорк оседлал Тучка
Снусмумрик оседлал Тучка
Мумми-папа оседлал Тучка
Фрекенснорк взлетает и пытается управлять тучкой.
Поворачивает.
Даёт задний ход.
Взлетает выше.
Взлетает выше.
Поворачивает.
Фрекенснорк кричит:"А ну давай! Гоп!"
Снусмумрик взлетает и пытается управлять тучкой.
Поворачивает.
Ускоряется.
Снусмумрик не справляется с управлением, падает и разбивается насмерть.

Выводы по работе:

Разобрался с классификацией исключений, вложенными, анонимными и локальными классами, научился применять их на практике.