Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант №0.15

*Выполнил:*

Студент группы P3113

Свиридов Дмитрий Витальевич

*Преподаватель:*

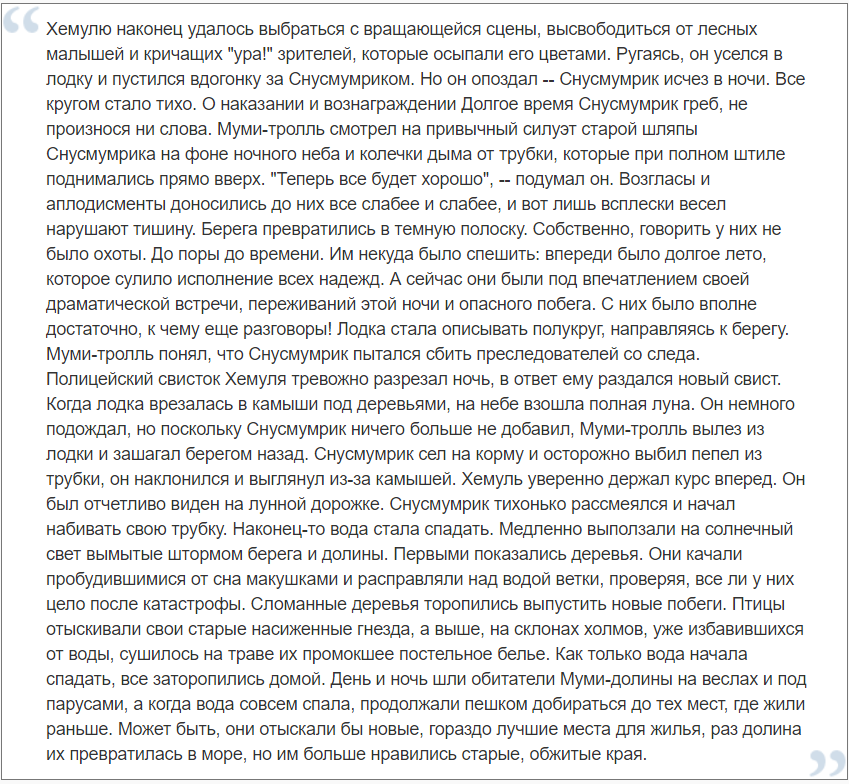
Письмак Алексей Евгеньевич



Санкт-Петербург, 2019

Диаграмма классов:



Описание предметной области:

Исходный код:

**HopesIsAlreadyFulfilled.java**

package utility;

public class HopesIsAlreadyFulfilled extends RuntimeException {

    @Override

    public String toString() {

        return "Мечты уже и так преисполнены!";

    }

}

**NoHopesForSummer.java**

package utility;

public class NoHopesForSummer extends Exception {

    @Override

    public String toString() {

        return "Лето может исполнять лишь мечты!";

    }

}

**Story.java**

package run;

import core.\*;

public class Story {

    public static void main(String[] args) {

*// TODO: Story*

    }

}

**SeasonAbstract.java**

package utility;

public abstract class SeasonAbstract implements SeasonInterface {

    private boolean lengthy;

    public SeasonAbstract(boolean lengthy) {

        this.lengthy = lengthy;

    }

    @Override

    public boolean isLengthy() {

        return lengthy;

    }

    @Override

    public void setLengthy(boolean lengthy) {

        this.lengthy = lengthy;

    }

}

**SeasonInterface.java**

package utility;

public interface SeasonInterface extends ThingInterface {

    SeasonType getType();

    boolean isLengthy();

    void setLengthy(boolean lengthy);

}

**SeasonType.java**

package utility;

public enum SeasonType {

    SUMMER,

    AUTUMN,

    WINTER,

    SPRING

}

**ThingInterface.java**

package utility;

public interface ThingInterface {

    String getName();

}

**Boat.java**

package core;

import utility.ThingInterface;

public class Boat implements ThingInterface {

private String name;

public Boat() {

name = "Лодка";

}

public Boat(String name) {

this.name = name;

}

public String semicircle() {

return this.toString() + " стала описывать полукруг, направляясь к берегу.";

}

public String crash() {

return this.toString() + " врезалась в камыши под деревьями.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Лодка '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Boat) {

return name.equals(((Boat) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

**Hopes.java**

package core;

import utility.ThingInterface;

import utility.HopesIsAlreadyFulfilled;

public class Hopes implements ThingInterface {

private String name;

private boolean fulfilled = false;

public Hopes() {

name = "Надежды";

}

public Hopes(String name) {

this.name = name;

}

public Hopes(String name, boolean fulfilled) {

this.name = name;

this.fulfilled = fulfilled;

}

public void fulfill() {

if (fulfilled) {

throw new HopesIsAlreadyFulfilled();

} else {

fulfilled = true;

}

}

public boolean isFulfilled() {

return fulfilled;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

String ending = fulfilled ? "(исполнены)" : "(не исполнены)";

return "Надежды '" + name + "' " + ending;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Hopes) {

return name.equals(((Hopes) obj).getName()) && fulfilled == ((Hopes) obj).isFulfilled();

}

return false;

}

@Override

public int hashCode() {

if (fulfilled) return name.hashCode() + name.length();

return name.hashCode();

}

}

**Shores.java**

package core;

import utility.ThingInterface;

public class Shores implements ThingInterface {

private String name;

private boolean hurry = false;

private boolean washed = true;

public Shores() {

name = "Берега";

}

public Shores(String name) {

this.name = name;

}

public Shores(String name, boolean hurry, boolean washed) {

this.name = name;

this.hurry = hurry;

this.washed = washed;

}

public String turnIntoLine() {

return this.toString() + " превратились в темную полоску.";

}

public String gotoSun() {

return this.toString() + " медленно выползали на солнечный свет.";

}

public String becomeImpressed() {

return this.toString() + " под впечатлением...";

}

public String becomeImpressed(String[] impressions) {

String result = this.toString() + " под впечатлением ";

for (int i=0; i<impressions.length; i++) {

if (i != 0) result += ", ";

result += impressions[i];

}

return result + ".";

}

public boolean isHurry() {

return hurry;

}

public void setHurry(boolean hurry) {

this.hurry = hurry;

}

public boolean isWashed() {

return washed;

}

public void setWashed(boolean washed) {

this.washed = washed;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

String ending;

String wash\_add = "";

if (washed) wash\_add = "не ";

if (hurry) {

ending = "(спешат, " + wash\_add + "вымыты штормом)";

} else {

ending = "(не спешат, " + wash\_add + "вымыты штормом)";

}

return "Берега '" + name + "' " + ending;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Shores) {

return name.equals(((Shores) obj).getName()) &&

hurry == ((Shores) obj).isHurry() &&

washed == ((Shores) obj).isWashed();

}

return false;

}

@Override

public int hashCode() {

int hash = name.hashCode();

if (hurry) hash += name.length();

if (washed) hash += 69;

return hash;

}

}

**Summer.java**

package core;

import utility.NoHopesForSummer;

import utility.SeasonAbstract;

import utility.SeasonType;

public class Summer extends SeasonAbstract {

private String name;

private final SeasonType TYPE = SeasonType.SUMMER;

public Summer() {

super(true);

name = "Лето";

}

public Summer(String name) {

super(true);

this.name = name;

}

public Summer(String name, boolean lengthy) {

super(lengthy);

this.name = name;

}

public String fulfillHopes(Hopes obj) throws NoHopesForSummer {

if (obj instanceof Hopes) {

obj.fulfill();

return this.toString() + " сулило исполнение всех надежд '" + obj.getName() + "'...";

}

throw new NoHopesForSummer();

}

@Override

public SeasonType getType() {

return TYPE;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

if (isLengthy()) return "Долгое лето '" + name + "'";

return "Лето '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Summer) {

return name.equals(((Summer) obj).getName()) && isLengthy() == ((Summer) obj).isLengthy();

}

return false;

}

    @Override

    public int hashCode() {

        if (isLengthy()) return name.hashCode() + name.length();

        return name.hashCode();

    }

}

**Talks.java**

package core;

import utility.ThingInterface;

public class Talks implements ThingInterface {

private String name;

public Talks() {

name = "Разговоры";

}

public Talks(String name) {

this.name = name;

}

public String notNeeded() {

return this.toString() + " теперь ни к чему.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Разговоры '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Talks) {

return name.equals(((Talks) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

**Birds.java**

package core;

import utility.ThingInterface;

public class Birds implements ThingInterface {

private String name;

public Birds() {

name = "Птицы";

}

public Birds(String name) {

this.name = name;

}

public String findNest() {

return this.toString() + " отыскивали свои старые насиженные гнезда.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Птицы '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Birds) {

return name.equals(((Birds) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

**Moon.java**

package core;

import utility.ThingInterface;

public class Moon implements ThingInterface {

private String name;

private boolean full = false;

public Moon() {

name = "Луна";

}

public Moon(String name) {

this.name = name;

}

public Moon(String name, boolean full) {

this.name = name;

this.full = full;

}

public String rise() {

return this.toString() + " взошла.";

}

public boolean isFull() {

return full;

}

public void setFull(boolean full) {

this.full = full;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

String ending = full ? "(полная)" : "(не полная)";

return "Луна '" + name + "' " + ending;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Moon) {

return name.equals(((Moon) obj).getName()) && full == ((Moon) obj).isFull();

}

return false;

}

@Override

public int hashCode() {

if (full) return name.hashCode() + name.length();

return name.hashCode();

}

}

**Valleys.java**

package core;

import utility.ThingInterface;

public class Valleys implements ThingInterface {

private String name;

private boolean washed = true;

public Valleys() {

name = "Долины";

}

public Valleys(String name) {

this.name = name;

}

public Valleys(String name, boolean washed) {

this.name = name;

this.washed = washed;

}

public String turnIntoSea() {

return this.toString() + " превратились в море.";

}

public String gotoSun() {

return this.toString() + " медленно выползали на солнечный свет.";

}

public boolean isWashed() {

return washed;

}

public void setWashed(boolean washed) {

this.washed = washed;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

String ending = washed ? "(вымыты штормом)" : "(не вымыты штормом)";

return "Долины '" + name + "' " + ending;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Valleys) {

return name.equals(((Valleys) obj).getName()) && washed == ((Valleys) obj).isWashed();

}

return false;

}

@Override

public int hashCode() {

if (washed) return name.hashCode() + name.length();

return name.hashCode();

}

}

**Water.java**

package core;

import utility.ThingInterface;

public class Water implements ThingInterface {

private String name;

private boolean raised = true;

public Water() {

name = "Вода";

}

public Water(String name) {

this.name = name;

}

public Water(String name, boolean raised) {

this.name = name;

this.raised = raised;

}

public String becomeFalling() {

return this.toString() + " стала спадать.";

}

public boolean isRaised() {

return raised;

}

public void setRaised(boolean raised) {

this.raised = raised;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

String ending = raised ? "(поднята)" : "(не поднята)";

return "Вода '" + name + "' " + ending;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Water) {

return name.equals(((Water) obj).getName()) && raised == ((Water) obj).isRaised();

}

return false;

}

@Override

public int hashCode() {

if (raised) return name.hashCode() + name.length();

return name.hashCode();

}

}

**Trees.java**

package core;

import utility.ThingInterface;

public class Trees implements ThingInterface {

private String name;

private class Crowns {

public String wave() {

return "Макушки деревьев '" + name + "' качаются.";

}

}

private class Branches {

public String spread() {

return "Ветки деревьев '" + name + "' расправились";

}

}

Crowns crowns = new Crowns();

Branches branches = new Branches();

public Trees() {

name = "Деревья";

}

public Trees(String name) {

this.name = name;

}

public String waveCrowns() {

return crowns.wave();

}

public String spreadBranches() {

return branches.spread();

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Деревья '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Trees) {

return name.equals(((Trees) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

**BrokenTrees.java**

package core;

public class BrokenTrees extends Trees {

public BrokenTrees() {

super();

}

public BrokenTrees(String name) {

super(name);

}

public String haste() {

return this.toString() + " торопились выпустить новые побеги.";

}

@Override

public String toString() {

return "Сломанные деревья '" + super.getName() + "'";

}

}

**Hemul.java**

package core;

import utility.ThingInterface;

public class Hemul implements ThingInterface {

private String name;

private boolean seen;

public static class PoliceWhistle implements ThingInterface {

private String name;

public PoliceWhistle() {

name = "Полицейский свисток";

}

public PoliceWhistle(String name) {

this.name = name;

}

public String cutNight() {

return this.toString() + " тревожно разрезал ночь.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Полицейский свисток '" + name + "' ";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof PoliceWhistle) {

return name.equals(((PoliceWhistle) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

public Hemul() {

name = "Хемуль";

}

public Hemul(String name) {

this.name = name;

}

public String getAwayFromStage() {

return this.toString() + " вырался с вращающейся сцены.";

}

public String escapeFromBadGuys() {

class BadGuys {};

BadGuys forestGuys = new BadGuys() {

@Override

public String toString() {

return "'Лесные малыши'";

}

};

BadGuys spectators = new BadGuys() {

public String voice() {

return this.toString() + " кричат 'Ура!'.";

}

public String throwFlowers(ThingInterface obj) {

return this.toString() + " осыпают цветами " + obj + ".";

}

@Override

public String toString() {

return "'Зрители'";

}

};

return this.toString() + " высвободился от " + forestGuys + " и " + spectators + ".";

}

public String seatIntoBoat(boolean angry) {

String ifAngry = angry ? ", ругаясь," : "";

return this.toString() + ifAngry + " уселся в лодку.";

}

public String seatIntoBoat() {

return seatIntoBoat(false);

}

public String catchIt(ThingInterface obj) {

return this.toString() + " пустился вдогонку за " + obj + ".";

}

public String getLate() {

return this.toString() + " опоздал.";

}

public String goForward() {

return this.toString() + " уверенно держал курс вперед.";

}

public boolean isSeen() {

return seen;

}

public void setSeen(boolean seen) {

this.seen = seen;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Хемуль '" + name + "' ";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Hemul) {

return name.equals(((Hemul) obj).getName()) && seen == ((Hemul) obj).isSeen();

}

return false;

}

@Override

public int hashCode() {

if (seen) return name.hashCode() + name.length();

return name.hashCode();

}

}}

public String cutNight() {

return this.toString() + " тревожно разрезал ночь.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Полицейский свисток '" + name + "' ";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof PoliceWhistle) {

return name.equals(((PoliceWhistle) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

public Hemul() {

name = "Хемуль";

}

public Hemul(String name) {

this.name = name;

}

public String getAwayFromStage() {

return this.toString() + " вырался с вращающейся сцены.";

}

public String escapeFromBadGuys() {

class BadGuys {};

BadGuys forestGuys = new BadGuys() {

@Override

public String toString() {

return "'Лесные малыши'";

}

};

BadGuys spectators = new BadGuys() {

public String voice() {

return this.toString() + " кричат 'Ура!'.";

}

public String toString() {

return "'Зрители'";

}

};

return this.toString() + " высвободился от " + forestGuys + " и " + spectators + ".";

}

public String seatIntoBoat(boolean angry) {

String ifAngry = angry ? ", ругаясь," : "";

return this.toString() + ifAngry + " уселся в лодку.";

}

public String seatIntoBoat() {

return seatIntoBoat(false);

}

public String catchIt(ThingInterface obj) {

return this.toString() + " пустился вдогонку за " + obj + ".";

}

public String getLate() {

return this.toString() + " опоздал.";

}

public String goForward() {

return this.toString() + " уверенно держал курс вперед.";

}

public boolean isSeen() {

return seen;

}

public void setSeen(boolean seen) {

this.seen = seen;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Хемуль '" + name + "' ";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Hemul) {

return name.equals(((Hemul) obj).getName()) && seen == ((Hemul) obj).isSeen();

}

return false;

}

@Override

public int hashCode() {

if (seen) return name.hashCode() + name.length();

return name.hashCode();

}

}

public String throwFlowers(ThingInterface obj) {

return this.toString() + " осыпают цветами " + obj + ".";

}

@Override

public String toString() {

return "'Зрители'";

}

};

return this.toString() + " высвободился от " + forestGuys + " и " + spectators + ".";

}

public String seatIntoBoat(boolean angry) {

String ifAngry = angry ? ", ругаясь," : "";

return this.toString() + ifAngry + " уселся в лодку.";

}

public String seatIntoBoat() {

return seatIntoBoat(false);

}

public String catchIt(ThingInterface obj) {

return this.toString() + " пустился вдогонку за " + obj + ".";

}

public String getLate() {

return this.toString() + " опоздал.";

}

public String goForward() {

return this.toString() + " уверенно держал курс вперед.";

}

public boolean isSeen() {

return seen;

}

public void setSeen(boolean seen) {

this.seen = seen;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Хемуль '" + name + "' ";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Hemul) {

return name.equals(((Hemul) obj).getName()) && seen == ((Hemul) obj).isSeen();

}

return false;

}

@Override

public int hashCode() {

if (seen) return name.hashCode() + name.length();

return name.hashCode();

}

}

@Override

public int hashCode() {

if (seen) return name.hashCode() + name.length();

return name.hashCode();

}

}

**Water.java**

package core;

import utility.ThingInterface;

public class Water implements ThingInterface {

private String name;

private boolean raised = true;

public Water() {

name = "Вода";

}

public Water(String name) {

this.name = name;

}

public Water(String name, boolean raised) {

this.name = name;

this.raised = raised;

}

public String becomeFalling() {

return this.toString() + " стала спадать.";

}

public boolean isRaised() {

return raised;

}

public void setRaised(boolean raised) {

this.raised = raised;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

String ending = raised ? "(поднята)" : "(не поднята)";

return "Вода '" + name + "' " + ending;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Water) {

return name.equals(((Water) obj).getName()) && raised == ((Water) obj).isRaised();

}

return false;

}

@Override

public int hashCode() {

if (seen) return name.hashCode() + name.length();

return name.hashCode();

}

}

**Mumi.java**

package core;

import utility.ThingInterface;

public class Mumi implements ThingInterface {

private String name;

public Mumi() {

name = "Муми-тролль";

}

public Mumi(String name) {

this.name = name;

}

public String watchAt() {

return this.toString() + " смотрел вдаль...";

}

public String watchAt(String[] itemsToWatch) {

String result = this.toString() + " смотрел на ";

for (int i=0; i<itemsToWatch.length; i++) {

if (i != 0) result += ", ";

result += itemsToWatch[i];

}

return result + ".";

}

public String thinkAbout(String thoughts) {

return this.toString() + " подумал: '" + thoughts + "'.";

}

public String consider(String thoughts) {

return this.toString() + " понял: '" + thoughts + "'.";

}

public String getOutOfBoat() {

return this.toString() + " вылез из лодки.";

}

public String goShores() {

return this.toString() + " зашагал берегом назад.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Муми-тролль '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Mumi) {

return name.equals(((Mumi) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Mumi) {

return name.equals(((Mumi) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

**Snusmumrik.java**

package core;

import utility.ThingInterface;

public class Snusmumrik implements ThingInterface {

// TODO: вынести трубку в отдельный вложенный класс

private String name;

public Snusmumrik() {

name = "Снусмумрик";

}

public Snusmumrik(String name) {

this.name = name;

}

public String hideInNight() {

return this.toString() + " исчез в ночи.";

}

public String oar(boolean silent) {

String ending = silent ? ", не произнося ни слова." : ".";

return this.toString() + " грёб" + ending;

}

public String oar() {

return oar(true);

}

public String seatOnShit() {

return this.toString() + " сел на корму.";

}

public String laugh(boolean lightly) {

String ifLaugh = lightly ? " тихонько" : "";

return this.toString() + ifLaugh + " рассмеялся.";

}

public String laugh() {

return laugh(true);

}

public String startFillingPipe() {

return this.toString() + " начал набивать трубку.";

}

public String cleanPipe(boolean carefully) {

String ifCarefully = carefully ? " осторожно" : "";

return this.toString() + ifCarefully + " выбил пепел из трубки.";

}

public String cleanPipe() {

return cleanPipe(true);

}

public String getDown() {

return this.toString() + " наклонился.";

}

public String lookOverReeds() {

return this.toString() + " выглянул из-за камышей.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Снусмумрик '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Snusmumrik) {

return name.equals(((Snusmumrik) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

} @Override

public String toString() {

return "Муми-тролль '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Mumi) {

return name.equals(((Mumi) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

Вывод:

public String startFillingPipe() {

return this.toString() + " начал набивать трубку.";

}

public String cleanPipe(boolean carefully) {

String ifCarefully = carefully ? " осторожно" : "";

return this.toString() + ifCarefully + " выбил пепел из трубки.";

}

public String cleanPipe() {

return cleanPipe(true);

}

public String getDown() {

return this.toString() + " наклонился.";

}

public String lookOverReeds() {

return this.toString() + " выглянул из-за камышей.";

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "Снусмумрик '" + name + "'";

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj instanceof Snusmumrik) {

return name.equals(((Snusmumrik) obj).getName());

}

return false;

}

@Override

public int hashCode() {

return name.hashCode();

}

}

Во время выполнения данной лабораторной работы я закрепил принципы SOLID, научился создавать собственные исключения и пользоваться вложенными, внутренними, локальными и анонимными классами. Не выспался.