Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерных технологий

Лабораторная работа №3

Вариант 213636546

Выполнила:

Павличенко Софья Алексеевна, Р3115

Проверил:

Вербовой Александр Александрович

Санкт-Петербург 2023г.

Оглавление

[Задание 3](#_Toc154036474)

[Диаграмма классов реализованной объектной модели 4](#_Toc154036475)

[Решение 4](#_Toc154036476)

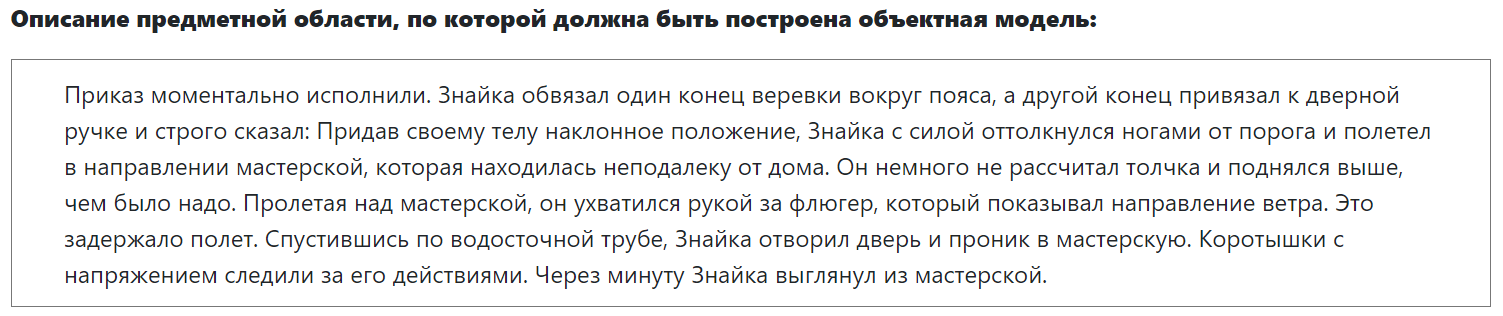
[Исходный код программы 4](#_Toc154036477)

[Результат работы программы 14](#_Toc154036478)

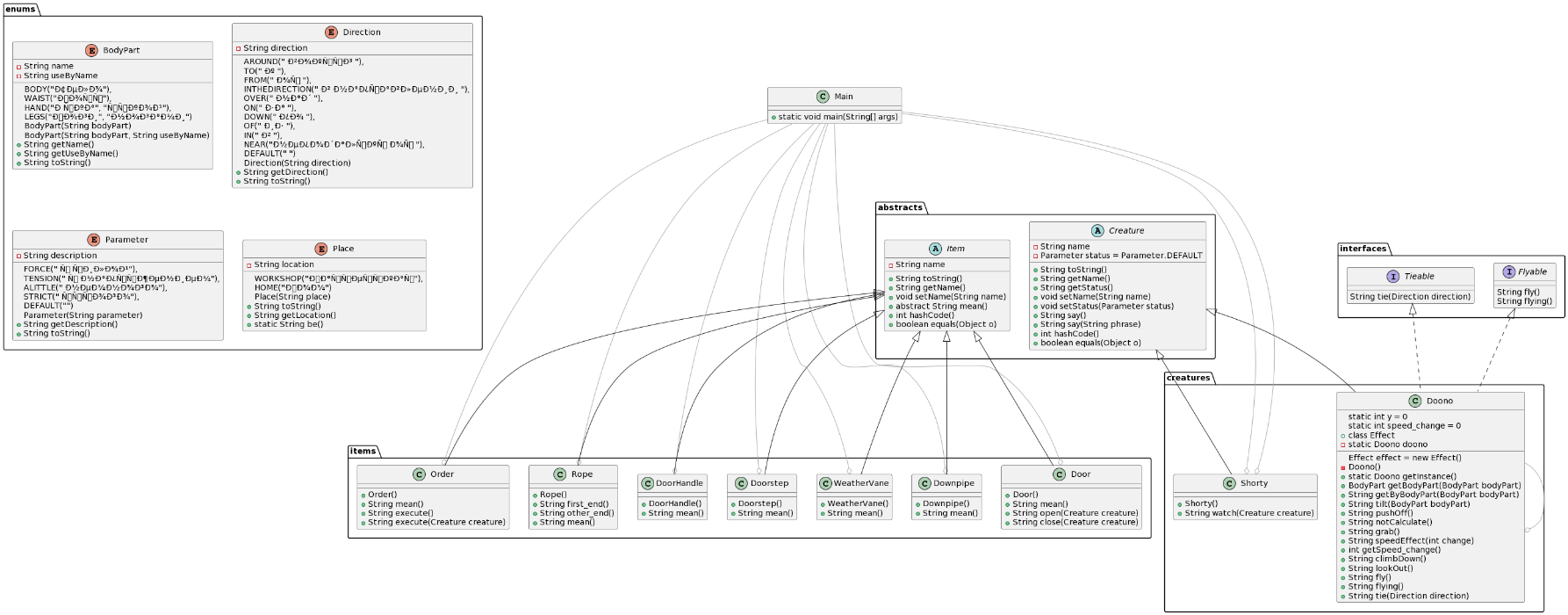
[Заключение 15](#_Toc154036479)

# Задание

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.



# Диаграмма классов реализованной объектной модели



# Решение

## Исходный код программы

Main.java

import creatures.\*;  
import enums.BodyPart;  
import enums.Direction;  
import enums.Parameter;  
import enums.Place;  
import items.\*;  
  
public class Main {б  
 public static void main(String[] args) {  
 Doono doono = Doono.*getInstance*();  
  
 Order order = new Order();  
 System.*out*.println(order.execute() + ".");  
  
 Rope rope = new Rope();  
 System.*out*.println(doono.tie(Direction.*AROUND*) + rope.first\_end() + Direction.*AROUND* + doono.getBodyPart(BodyPart.*WAIST*) + ".");  
  
 DoorHandle doorHandle = new DoorHandle();  
 System.*out*.println(doono.tie(Direction.*TO*) + rope.other\_end() + Direction.*TO* + doorHandle + ".");  
  
 doono.setStatus(Parameter.*STRICT*);  
 System.*out*.println(doono.say() + ".");  
 doono.setStatus(Parameter.*DEFAULT*);  
  
 System.*out*.println(doono.tilt(BodyPart.*BODY*) + ".");  
  
 Doorstep doorstep = new Doorstep();  
 doono.setStatus(Parameter.*FORCE*);  
 System.*out*.println(doono.pushOff() + doono.getByBodyPart(BodyPart.*LEGS*) + Direction.*FROM* + doorstep + ".");  
 doono.setStatus(Parameter.*DEFAULT*);  
  
 System.*out*.println(doono.fly() + Direction.*INTHEDIRECTION* + Place.*WORKSHOP* + ".");  
 System.*out*.println(Place.*WORKSHOP* + Place.*be*() + Direction.*NEAR* + Place.*HOME* + ".");  
  
 doono.setStatus(Parameter.*ALITTLE*);  
 System.*out*.println(doono.notCalculate() + ".");  
 doono.setStatus(Parameter.*DEFAULT*);  
  
 System.*out*.println(doono.fly() + ".");  
  
 WeatherVane weatherVane = new WeatherVane();  
 System.*out*.println(doono.flying() + Direction.*OVER* + Place.*WORKSHOP* + ", " + doono.grab() + doono.getByBodyPart(BodyPart.*HAND*) + Direction.*ON* + weatherVane + ",");  
 System.*out*.println(weatherVane + weatherVane.mean() + ".");  
  
 System.*out*.println(doono.speedEffect(doono.getSpeed\_change()) + ".");  
  
 Downpipe downpipe = new Downpipe();  
 System.*out*.println(doono.climbDown() + Direction.*DOWN* + downpipe + ".");  
  
 Door door = new Door();  
 System.*out*.println(door.open(doono) + ".");  
  
 System.*out*.println(doono.toString() + Direction.*IN* + Place.*WORKSHOP* + ".");  
  
 Shorty[] shorties = new Shorty[16];  
 for (int i = 0; i < shorties.length; i++) {  
 shorties[i] = new Shorty();  
 shorties[i].setStatus(Parameter.*TENSION*);  
 System.*out*.println(shorties[i].watch(doono) + ".");  
 }  
  
 System.*out*.println(doono.lookOut() + Direction.*OF* + Place.*WORKSHOP* + ".");  
 }  
}

**Пакет abstracts**

Creature.java

package abstracts;  
  
import enums.Parameter;  
  
public abstract class Creature {  
 private String name;  
 private Parameter status = Parameter.*DEFAULT*;  
  
 @Override  
 public String toString() {  
 return name;  
 }  
 public String getName() {  
 return name;  
 }  
 public String getStatus() {  
 return status.toString();  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
 public void setStatus(Parameter status) {  
 this.status = status;  
 }  
  
 public String say() {  
 return this + this.getStatus() + " сказал";  
 }  
 public String say(String phrase) {  
 return this + this.getStatus() + " сказал: \"" + phrase + "\"";  
 }  
  
 @Override  
 public int hashCode() {  
 int result = name.hashCode();  
 result = 31 \* result + status.hashCode();  
 return result;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Creature that = (Creature) o;  
  
 return this.name.equals(that.name) && this.status == that.status;  
 }  
}

Item.java

package abstracts;  
  
public abstract class Item {  
 private String name;  
  
 @Override  
 public String toString() {  
 return name;  
 }  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
 public abstract String mean();  
  
 @Override  
 public int hashCode() {  
 int result = name.hashCode();  
 result = 31 \* result + this.mean().hashCode();  
 return result;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Item that = (Item) o;  
  
 return this.name.equals(that.name) && this.mean().equals(that.mean());  
 }  
}

**Пакет creatures**

Doono.java

package creatures;  
import abstracts.\*;  
import enums.\*;  
import interfaces.\*;  
  
public class Doono extends Creature implements Flyable, Tieable {  
 static int *y* = 0;  
 static int *speed\_change* = 0;  
  
 public class Effect {  
 String description = "";  
 public String getDescription() {  
 return description;  
 }  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 @Override  
 public String toString() {  
 return description;  
 }  
 }  
 Effect effect = new Effect();  
  
 private static Doono *doono*;  
 private Doono() {  
 super.setName("Знайка");  
 }  
  
 public static Doono getInstance() {  
 if (*doono* == null) {  
 *doono* = new Doono();  
 }  
 return *doono*;  
 }  
  
  
 public BodyPart getBodyPart(BodyPart bodyPart) {  
 return bodyPart;  
 }  
 public String getByBodyPart(BodyPart bodyPart) {  
 return bodyPart.getUseByName();  
 }  
 public String tilt(BodyPart bodyPart) {  
 return this + this.getStatus() + " наклонил " + getBodyPart(bodyPart);  
 }  
 public String pushOff() {  
 *y* += 1;  
 *speed\_change* = Math.*max*(*speed\_change*, 0) + 1;  
 effect.setDescription("Толчок");  
 return this + this.getStatus() + " оттолкнулся ";  
 }  
 public String notCalculate() {  
 *y* += 1;  
 *speed\_change* = Math.*max*(*speed\_change*, 0) + 1;  
 return this + this.getStatus() + " не рассчитал " + effect;  
 }  
 public String grab() {  
 *speed\_change* = Math.*min*(*speed\_change*, 0) - 1;  
 this.speedEffect(-1);  
 return this + this.getStatus() + " ухватился ";  
 }  
 public String speedEffect(int change) {  
 if (change > 0) return "Это ускорило " + effect;  
 else if (change < 0) return "Это задержало " + effect;  
 else return "";  
 }  
 public int getSpeed\_change() {  
 return *speed\_change*;  
 }  
 public String climbDown() {  
 return this + this.getStatus() + " спустился";  
 }  
 public String lookOut() {  
 return this + this.getStatus() + " выглянул";  
 }  
 @Override  
 public String fly() {  
 return this + this.getStatus() + switch (*y*) {  
 case 0 -> "";  
 case 1 -> " полетел";  
 case 2 -> " поднялся выше, чем было надо";  
 default -> "лететь";  
 };  
 }  
 @Override  
 public String flying() {  
 if (*y* > 0) {  
 effect.setDescription("полёт");  
 return this + this.getStatus() + " пролетал";  
 }  
 return null;  
 }  
  
 @Override  
 public String tie(Direction direction) {  
 return this + this.getStatus() + switch (direction) {  
 case *AROUND* -> " обвязал ";  
 case *TO* -> " привязал ";  
 default -> "вязать";  
 };  
 }  
}

Shorty.java

package creatures;  
import abstracts.Creature;  
  
public class Shorty extends Creature {  
 public Shorty() {  
 super.setName("Коротышка");  
 }  
  
 public String watch(Creature creature) {  
 return this + this.getStatus() + " следил за " + creature;  
 }  
}

**Пакет enums**

BodyPart.java

package enums;  
  
public enum BodyPart {  
 *BODY*("Тело"),  
 *WAIST*("Пояс"),  
 *HAND*("Рука", "рукой"),  
 *LEGS*("Ноги", "ногами");  
  
 private String name;  
 private String useByName;  
 BodyPart(String bodyPart) {  
 name = bodyPart;  
 }  
  
 BodyPart(String bodyPart, String useByName) {  
 name = bodyPart;  
 this.useByName = useByName;  
 }  
 public String getName() {  
 return name;  
 }  
 public String getUseByName() {  
 return useByName;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
}

Direction.java

package enums;  
public enum Direction {  
 AROUND(" вокруг "),  
 TO(" к "),  
 FROM(" от "),  
 INTHEDIRECTION(" в направлении "),  
 OVER(" над "),  
 ON(" за "),  
 DOWN(" по "),  
 OF(" из "),  
 *IN*(" в "),  
 *NEAR*("неподалёку от "),  
 *DEFAULT*(" ");  
  
 private String direction;  
  
 Direction(String direction) {  
 this.direction = direction;  
 }  
  
 public String getDirection() {  
 return direction;  
 }  
  
 @Override  
 public String toString() {  
 return direction;  
 }  
}

Parameter.java

package enums;  
  
public enum Parameter {  
 *FORCE*(" с силой"),  
 *TENSION*(" с напряжением"),  
 *ALITTLE*(" немного"),  
 STRICT(" строго"),  
 DEFAULT("");  
  
 private String description;  
 Parameter(String parameter) {  
 description = parameter;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 @Override  
 public String toString() {  
 return description;  
 }  
}

Place.java

package enums;  
  
public enum Place {  
 *WORKSHOP*("Мастерская"),  
 *HOME*("Дом");  
  
 private String location;  
 Place(String place) {  
 location = place;  
 }  
  
 @Override  
 public String toString() {  
 return location;  
 }  
  
 public String getLocation() {  
 return location;  
 }  
  
 public static String be() { return " находится ";}  
}

**Пакет interfaces**

Flyable.java

package interfaces;  
  
public interface Flyable {  
 String fly();  
 String flying();  
}

Tieable.java

package interfaces;  
  
import enums.Direction;  
  
public interface Tieable {  
 String tie(Direction direction);  
}

**Пакет items**

Door.java

package items;  
import abstracts.Creature;  
import abstracts.Item;  
public class Door extends Item {  
 public Door() {  
 super.setName("Дверь");  
 }  
  
 @Override  
 public String mean() {  
 return "";  
 }  
  
 public String open(Creature creature) {  
 return creature + " отворил дверь";  
 }  
  
 public String close(Creature creature) {  
 return creature + " закрыл дверь";  
 }  
  
}

Order.java

package items;  
import abstracts.Creature;  
import abstracts.Item;  
public class Order extends Item {  
 public Order() {  
 super.setName("Приказ");  
 }  
  
 @Override  
 public String mean() {  
 return "";  
 }  
  
 public String execute() {  
 return this + " исполнен";  
 }  
  
 public String execute(Creature creature) {  
 return creature + " исполнил " + this;  
 }  
}

Rope.java

package items;  
import abstracts.Item;  
public class Rope extends Item {  
 public Rope() {  
 super.setName("Верёвка");  
 }  
 public String first\_end() {  
 return "один конец верёвки";  
 }  
 public String other\_end() {  
 return "другой конец верёвки";  
 }  
  
 @Override  
 public String mean() {  
 return "";  
 }  
}

WeatherVane.java

package items;  
import abstracts.Item;  
public class WeatherVane extends Item{  
 public WeatherVane() {  
 super.setName("Флюгер");  
 }  
 @Override  
 public String mean() {  
 return " показывает направление ветра";  
 }  
}

## Результат работы программы

Приказ исполнен.

Знайка обвязал один конец верёвки вокруг Пояс.

Знайка привязал другой конец верёвки к Дверная ручка.

Знайка строго сказал.

Знайка наклонил Тело.

Знайка с силой оттолкнулся ногами от Порог.

Знайка полетел в направлении Мастерская.

Мастерская находится неподалёку от Дом.

Знайка немного не рассчитал Толчок.

Знайка поднялся выше, чем было надо.

Знайка пролетал над Мастерская, Знайка ухватился рукой за Флюгер,

Флюгер показывает направление ветра.

Это задержало полёт.

Знайка спустился по Водосточная труба.

Знайка отворил дверь.

Знайка в Мастерская.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Коротышка с напряжением следил за Знайка.

Знайка выглянул из Мастерская.

# Заключение

В результате выполнения лабораторной работы я познакомилась с принципами SOLID и STUPID, научилась работать с абстрактными классами, интерфейсами, перечисляемыми типами, узнала о некоторых методах класса Object и научилась их переопределять.