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

Факультет программной инженерии и компьютерных технологий

Лабораторная работа №4

Вариант 213636546,6

Выполнила:

Павличенко Софья Алексеевна, Р3115

Проверил:

Вербовой Александр Александрович

Санкт-Петербург 2023г.

Оглавление

[Задание 3](#_Toc154232119)

[Диаграмма классов реализованной объектной модели 4](#_Toc154232120)

[Решение 5](#_Toc154232121)

[Исходный код программы 5](#_Toc154232122)

[Результат работы программы 18](#_Toc154232123)

[Заключение 21](#_Toc154232124)

# 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 abstracts.Creature;  
import creatures.\*;  
import enums.\*;  
import exceptions.\*;  
import items.\*;  
  
import java.util.ArrayList;  
  
public class Main {  
 public static void main(String[] args) {  
  
 ArrayList<Creature> creatures = new ArrayList<>();  
  
 Bendum bendum = Bendum.*getInstance*();  
 creatures.add(bendum);  
 System.*out*.println(bendum.notRespond() + ".");  
  
 Doono doono = Doono.*getInstance*();  
 creatures.add(doono);  
  
 System.*out*.println(doono.lookOut() + Direction.*IN* + Place.*HALL* + ".");  
  
 Twistum twistum = Twistum.*getInstance*();  
 creatures.add(twistum);  
  
 System.*out*.println(doono.hear(twistum) + ".");  
  
 Shorty[] shorties = new Shorty[13];  
 for (int i = 0; i < shorties.length; i++) {  
 shorties[i] = new Shorty();  
 creatures.add(shorties[i]);  
 }  
  
 if (creatures.size() < 4) {  
 throw new NotEnoughCreaturesException("Ошибка: Слишком мало действующих лиц!");  
 }  
  
 for (Creature creature : creatures) {  
 System.*out*.println(creature.takeAlarm() + ".");  
 System.*out*.println(creature.rush() + Direction.*TO* + Place.*EXIT* + ".");  
 }  
  
 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.getWaist() + ".");  
  
 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.tiltBody() + ".");  
  
 Doorstep doorstep = new Doorstep();  
 doono.setStatus(Parameter.*FORCE*);  
 System.*out*.println(doono.pushOffWithLegs() + 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* + ".");  
  
 try {  
 System.*out*.println(doono.Calculate());  
 }  
 catch (NotCalculateException e) {  
 System.*out*.println("Ошибка: " + e.getMessage() + ".");  
 }  
 finally {  
 System.*out*.println(doono.fly() + ".");  
 }  
  
 WeatherVane weatherVane = new WeatherVane();  
 System.*out*.println(doono.flying() + Direction.*OVER* + Place.*WORKSHOP* + ", " + doono.grabWithHand() + Direction.*WITH* + 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.*DUP* + downpipe + ".");  
  
 Door door = new Door();  
 System.*out*.println(door.open(doono) + ".");  
  
 System.*out*.println(doono.toString() + Direction.*IN* + Place.*WORKSHOP* + ".");  
  
 for (Shorty shorty : shorties) {  
 shorty.setStatus(Parameter.*TENSION*);  
 System.*out*.println(shorty.watch(doono) + ".");  
 shorty.setStatus(Parameter.*DEFAULT*);  
 }  
  
 System.*out*.println(doono.lookOut() + Direction.*OF* + Place.*WORKSHOP* + ".");  
  
 System.*out*.println(doono.jump() + Direction.*TO* + Place.*PAVILION* + ".");  
 System.*out*.println(doono.lookIn() + Direction.*INSIDE* + ".");  
  
 System.*out*.println(bendum.notBe() + Direction.*IN* + Place.*PAVILION* + ".");  
  
 for (Shorty shorty : shorties) {  
 System.*out*.println(shorty.pull() + rope + ".");  
 }  
 System.*out*.println(doono.toString() + Direction.*IN* + Place.*HOME* + ".");  
  
 doono.setStatus(Parameter.*INSTANTLY*);  
 System.*out*.println(doono.climbUp() + Direction.*DUP* + downpipe + Direction.*ON* + Place.*ROOF* + ".");  
 doono.setStatus(Parameter.*DEFAULT*);  
  
 System.*out*.println(doono.wantLookAround() + ".");  
  
 System.*out*.println(doono.lostControl());  
 doono.setStatus(Parameter.*NOTAFRAID*);  
 System.*out*.println(doono + ".");  
 doono.setStatus(Parameter.*DEFAULT*);  
  
 System.*out*.println(doono.know(shorties[(int)(Math.*random*() \* (creatures.size()))].canPull() + Direction.*ON* + rope + " " + doono + Direction.*BACK* + "."));  
 }  
}

**Пакет abstracts**

Creature.java

package abstracts;  
package abstracts;  
  
import creatures.Bendum;  
import enums.Parameter;  
import enums.Place;  
  
public abstract class Creature {  
 private String name;  
 private Parameter status = Parameter.*DEFAULT*;  
  
 @Override  
 public String toString() {  
 return name + status;  
 }  
 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 toString() + " сказал";  
 }  
 public String say(String phrase) {  
 return toString() + " сказал: \"" + phrase + "\"";  
 }  
   
 public String takeAlarm() {  
 return toString() + " всполошился";  
 }  
 public String rush() {  
 return toString() + " бросился";  
 }  
  
 @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 exceptions.NotCalculateException;  
import interfaces.\*;  
  
public class Doono extends Creature implements Flyable {  
 private static Doono *doono*;  
 private Doono() {  
 setName("Знайка");  
 }  
  
 public static Doono getInstance() {  
 if (*doono* == null) {  
 *doono* = new Doono();  
 }  
 return *doono*;  
 }  
  
 static int *y* = 0;  
 static int *speedChange* = 0;  
  
 static 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();  
  
 public class BodyPart {  
 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;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
 public void setUseByName(String useByName) {  
 this.useByName = useByName;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
 }  
  
 public String grabWithHand() {  
 *speedChange* = Math.*min*(*speedChange*, 0) - 1;  
 speedEffect(-1);  
 BodyPart hand = new BodyPart("Рука", "рукой");  
 return toString() + " ухватился " + hand.getUseByName();  
 }  
  
 public String tiltBody() {  
 BodyPart body = new BodyPart("Тело");  
 return toString() + " наклонил " + body;  
 }  
  
 public String getWaist() {  
 BodyPart waist = new BodyPart("Пояс");  
 return waist.toString();  
 }  
 public String pushOffWithLegs() {  
 *y* += 1;  
 *speedChange* = Math.*max*(*speedChange*, 0) + 1;  
 effect.setDescription("Толчок");  
 BodyPart legs = new BodyPart("Нога", "ногами");  
 return toString() + " оттолкнулся " + legs.getUseByName();  
 }  
  
 public String Calculate() throws NotCalculateException {  
 *y* += 1;  
 *speedChange* = Math.*max*(*speedChange*, 0) + 1;  
 if (*y* > 1) {  
 throw new NotCalculateException(toString() + switch (*y*) {  
 case(2) -> Parameter.*ALITTLE* + " не рассчитал " + effect;  
 case(3) -> " не рассчитал " + effect;  
 default -> "сильно ошибся в расчётах " + effect;  
 });  
 }  
 return toString() + "верно рассчитал " + effect;  
 }  
 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 *speedChange*;  
 }  
 public String climbDown() {  
 *y* -= 1;  
 return toString() + " спустился";  
 }  
 public String climbUp() {  
 *y* += 1;  
 return toString() + " вскарабкался";  
 }  
 public String lookOut() {  
 return toString() + " выглянул";  
 }  
 public String lookIn() {  
 return toString() + " заглянул";  
 }  
  
 public String jump() {  
 *y* += 1;  
 *y* -= 1;  
 return toString() + " прыгнул";  
 }  
  
 public String wantLookAround() {  
 return toString() + " хотел оглянуться по сторонам";  
 }  
  
 public String hear(Creature creature) {  
 return toString() + " услыхал слова " + creature;  
 }  
  
 public String know(String knowledge) {  
 return toString() + " знал, что " + knowledge;  
 }  
 @Override  
 public String fly() {  
 return toString() + 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 toString() + " пролетал";  
 }  
 return null;  
 }  
  
 Tieable tieable = new Tieable() {  
 @Override  
 public String tie(Direction direction) {  
 return *doono*.toString() + switch (direction) {  
 case *AROUND* -> " обвязал ";  
 case *TO* -> " привязал ";  
 default -> "вязать";  
 };  
 }  
 };  
  
 public String tie(Direction direction) {  
 return tieable.tie(direction);  
 }  
  
 public String lostControl() {  
 class GustOfWind {  
 private String name = "Порыв ветра";  
 private String status;  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 public String blow() {  
 return toString() + " налетел";  
 }  
 public String blowOff(Creature creature, Place place) {  
 return toString() + " сдул c " + place + " " + creature;  
 }  
 public String carryAway(Creature creature) {  
 return toString() + " понёс в сторону " + creature;  
 }  
 }  
  
 GustOfWind gustOfWind = new GustOfWind();  
 gustOfWind.status = "Неожиданно ";  
 return gustOfWind.status + gustOfWind.blow() + ".\n" + gustOfWind.blowOff(*doono*, Place.*ROOF*) + ".\n" + gustOfWind.carryAway(*doono*) + ".";  
 }  
}

Shorty.java

package creatures;  
import abstracts.Creature;  
  
public class Shorty extends Creature {  
 public Shorty() {  
 setName("Коротышка");  
 }  
  
 public String watch(Creature creature) {  
 return toString() + " следил за " + creature;  
 }  
   
 public String pull() {  
 return toString() + " принялся тянуть ";  
 }  
  
 public String canPull() {  
 return toString() + " может притянуть";  
 }  
}

Bendum.java

package creatures;  
import abstracts.Creature;  
  
public class Bendum extends Creature {  
 private static Bendum *bendum*;  
 private Bendum() {  
 setName("Винтик");  
 }  
  
 public static Bendum getInstance() {  
 if (*bendum* == null) {  
 *bendum* = new Bendum();  
 }  
 return *bendum*;  
 }  
  
 public String notRespond() {  
 return toString() + " не отзывался";  
 }  
 public String notBe() {  
 return toString() + " не был";  
 }  
}

Twistum.java

package creatures;  
import abstracts.Creature;  
  
public class Twistum extends Creature {  
 private static Twistum *twistum*;  
 private Twistum() {  
 setName("Шпунтик");  
 }  
   
 public static Twistum getInstance() {  
 if (*twistum* == null) {  
 *twistum* = new Twistum();  
 }  
 return *twistum*;  
 }  
}

**Пакет enums**

Direction.java

package enums;  
public enum Direction {  
 *AROUND*(" вокруг "),  
 *TO*(" к "),  
 *FROM*(" от "),  
 *INTHEDIRECTION*(" в направлении "),  
 *OVER*(" над "),  
 *WITH*(" за "),  
 *DUP*(" по "),  
 *ON*(" на "),  
 *OF*(" из "),  
 *IN*(" в "),  
 *NEAR*("неподалёку от "),  
 *INSIDE*(" внутрь"),  
 *BACK*(" обратно"),  
 *DEFAULT*(" ");  
  
 private final 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"),  
 *INSTANTLY*(" мгновенно"),  
 *NOTAFRAID*(" не испуганный"),  
 *DEFAULT*("");  
  
 private final 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"),  
 *HALL*("Коридор"),  
 *EXIT*("Выход"),  
 *PAVILION*("Беседка"),  
 *ROOF*("Крыша");  
  
 private final 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() {  
 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";  
 }  
  
}

DoorHandle.java

package items;  
import abstracts.Item;  
public class DoorHandle extends Item{  
 public DoorHandle() {  
 setName("Дверная ручка");  
 }  
 @Override  
 public String mean() {  
 return "";  
 }  
}

Doorstep.java

package items;  
import abstracts.Item;  
public class Doorstep extends Item {  
 public Doorstep() {  
 setName("Порог");  
 }  
  
 @Override  
 public String mean() {  
 return "";  
 }  
  
}

Downpipe.java

package items;  
import abstracts.Item;  
  
public class Downpipe extends Item {  
 public Downpipe() {  
 setName("Водосточная труба");  
 }  
 @Override  
 public String mean() {  
 return "";  
 }  
}

Order.java

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

Rope.java

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

**Пакет exceptions**

NotCalculateException.java

package exceptions;  
  
public class NotCalculateException extends Exception{  
 public NotCalculateException(String message) {  
 super(message);  
 }  
}

NotEnoughCreaturesException.java

package exceptions;  
  
public class NotEnoughCreaturesException extends RuntimeException{  
 public NotEnoughCreaturesException(String message) {  
 super(message);  
 }  
}

## Результат работы программы

Винтик не отзывался.

Знайка выглянул в Коридор.

Знайка услыхал слова Шпунтик.

Винтик всполошился.

Винтик бросился к Выход.

Знайка всполошился.

Знайка бросился к Выход.

Шпунтик всполошился.

Шпунтик бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

Коротышка всполошился.

Коротышка бросился к Выход.

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.

Ошибка: 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.

Знайка выглянул из Мастерская.

Знайка прыгнул к Беседка.

Знайка заглянул внутрь.

Винтик не был в Беседка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Коротышка принялся тянуть Верёвка.

Знайка в Дом.

Знайка мгновенно вскарабкался по Водосточная труба на Крыша.

Знайка хотел оглянуться по сторонам.

Неожиданно Порыв ветра налетел.

Порыв ветра сдул c Крыша Знайка.

Порыв ветра понёс в сторону Знайка.

Знайка не испуганный.

Знайка знал, что Коротышка может притянуть на Верёвка Знайка обратно.

# Заключение

В результате выполнения лабораторной работы я научилась создавать static и non-static вложенные, анонимные и локальные классы, узнала об исключениях в Java и научилась реализовывать собственные и обрабатывать их.