МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА № 4

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант №456345.5

Выполнил:
Студент группы P3118
Шульга Артём Игоревич
Преподаватель:
Сорокин Роман Борисович

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2021

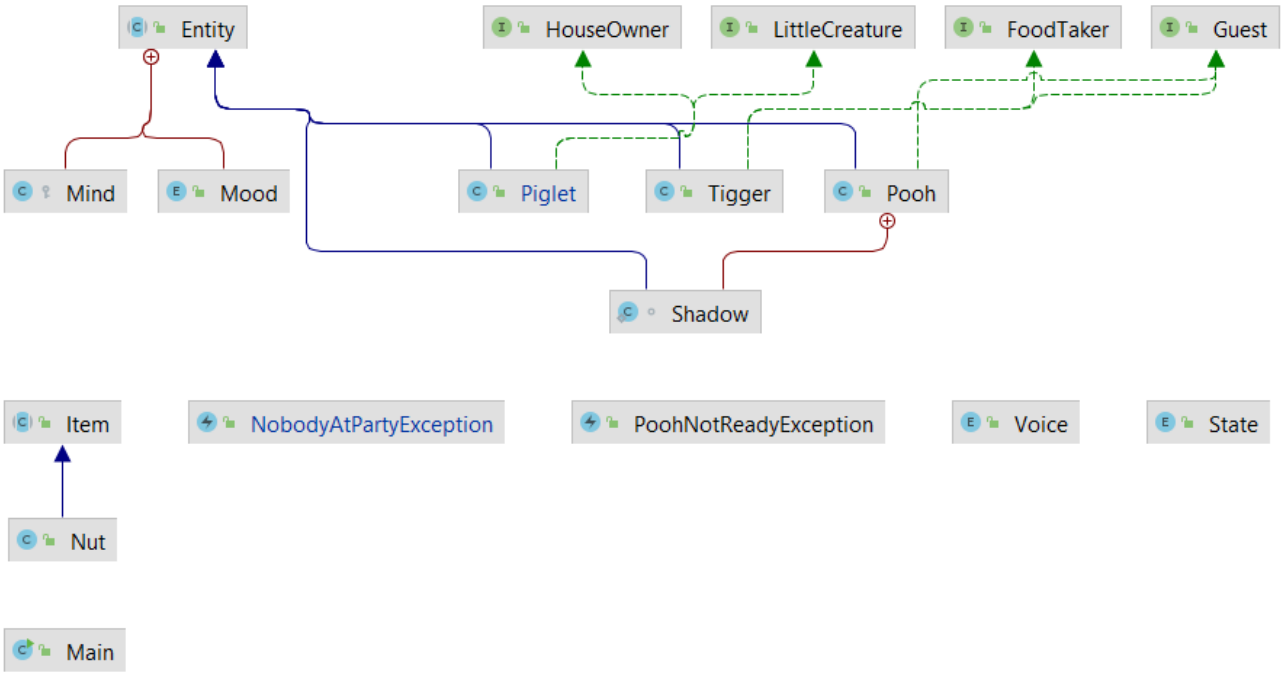
Задание

Введите вариант: 456345.5

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

А потом он сказал очень решительным голосом: Сказать по совести, Винни-Пуху это было довольно приятно, и он поспешно сообщил Тигре, что, как только он, Пух, справится со своим завтраком, они пойдут в гости к Пятачку, и, может быть, он угостит их желудями. И вот после завтрака они отправились в гости к Пятачку, и Пух по дороге объяснял, что Пятачок -- очень Маленькое Существо и не любит, когда на него наскакивают, так что он, Пух. просит Тигру не очень распрыгиваться для первого знакомства, а Тигра, который всю дорогу прятался за деревьями, то вдруг выскакивал из засады, стараясь поймать тень Пуха, когда она не смотрела, отвечал, что Тигры наскакивают только до завтрака, а едва они съедят немного желудей, они становятся Тихими и Вежливыми. Так они незаметно дошли до дверей Пятачка и постучали. Пятачок подвинул корзинку с желудями Тигре и сказал "Угощайтесь, пожалуйста", а сам крепко прижался к Пуху и, почувствовав себя гораздно храбрее, сказал: "Так ты Тигра? Ну-ну!" -- почти веселым голосом. Но Тигра ничего не ответил, потому что рот у него был набит желудями... Он долго и громко жевал их, а потом сказал: А когда Пух и Пятачок спросили: "Что, что?" -- он сказал:

Диаграмма реализованных классов



Исходный код

```
// Main.java

package com.company;

public class Main {
    public static void main(String[] args) {
        Pooh pooh = new Pooh("Пух",State.AT_HOUSE);
        Tigger tigger = new Tigger("Тигра",State.AT_HOUSE);
        Piglet piglet = new Piglet("Пятачок",State.AT_HOUSE);

        tigger.sayAboutProblem();
        pooh.suggest(tigger);

        System.out.println();

        int i=0;
        while(true){
            try {
                if(i == 2){
                    System.out.println("Пух психует и быстро доедает");
                    pooh.setState(State.AT_HOUSE,false,true);
                }
                tigger.tryToCall(pooh);
                break;
            } catch (Throwable t) {
                System.out.println(t.getMessage());
            }
            finally {
                i++;
            }
        }

        System.out.println();

        pooh.startJourneyWith(tigger,"дом Пятачка");

        System.out.println();

        pooh.tellAboutLittleCreature(tigger,piglet);
        tigger.playWithShadow(pooh.getShadow());

        System.out.println();

        pooh.endJourneyWith(tigger,piglet);

        System.out.println();

        piglet.giveFood(tigger);
        piglet.hug(pooh);
        piglet.speak("Так ты Тигра? Ну-ну!");
        tigger.speak("Да да я");

        tigger.tryToFinishEating();
        tigger.speak("Да, это я");

        Entity friends = new Entity("Пух и Пятачок", State.STAY) {
            @Override
            public void speak(String phrase) {
                System.out.println(name+" говорят: "+phrase);
            }
        };
        friends.speak("Что-что?");

        tigger.getMind().setVoice(Voice.NORMAL);
```

```
        tigger.speak("Я ЕСТЬ ТИГРА");
    }
}


// Entity.java

package com.company;

public abstract class Entity {
    protected final String name;
    private State state;
    private final Mind mind;
    Entity(final String name, final State state){
        this.name = name;
        this.state = state;
        mind = new Mind(); // singleton
    }
    protected final Mind getMind(){
        return mind;
    }
    protected final void setState(State state){
        setState(state,true,true);
    }
    protected final void setState(State state,boolean withStart,boolean withEnd){
        if(this.state.endState() != null && withEnd)
            System.out.println("*"+name+" "+this.state.endState()+"*");
        this.state = state;
        if(state.startState() != null && withStart)
            System.out.println("*"+name+" "+state.startState()+"*");
    }
    public final State getState(){
        return state;
    }
    public void speak(String phrase){
        if(mind.getVoice().isSilence)
            System.out.println(name+" "+ mind.getVoice().getVoicePhrase());
        else
            System.out.println(name+" "+ mind.getVoice().getVoicePhrase()+": "+phrase);
    }

    @Override
    public final String toString(){
        return name;
    }
    protected class Mind {
        private Voice voice;
        protected Mood mood;

        private Mind(){
            voice = Voice.NORMAL;
            mood = Mood.NORMAL;
        }

        public final Mood getMood() {
            return mood;
        }

        protected final void setVoice(Voice voice) {
            System.out.println("*голос у "+Entity.this.name+" меняется на
"+voice.getName()+"*");
            this.voice = voice;
        }
        private Voice getVoice(){
            return voice;
        }

        protected final void setMood(Mood mood) {
```

```
            System.out.println("*настроение у "+Entity.this.name+" меняется на
"+mood.getName()+"*");
        }
    }
    public enum Mood{
        NORMAL("нормальный"),
        HAPPY("довольный"),
        FEAR("испуганный");

        private final String name;
        Mood(final String name){
            this.name = name;
        }
        private String getName(){
            return name;
        }
    }

}


// Item.java

package com.company;

public abstract class Item {
    protected final String name;
    Item(String name){
        this.name = name;
    }
    public abstract void changeOwner(Entity entity);
}


// Nut.java

package com.company;

public class Nut extends Item{
    Nut(){
        super("орехи");
    }
    @Override
    public void changeOwner(Entity entity) {
        System.out.println(entity.name+" берёт "+this.name + " и начинает есть");
        entity.getMind().setVoice(Voice.CLOGGED_WITH_ACORNS);
    }
}


// Pooh.java

package com.company;

public class Pooh extends Entity implements Guest{
    private final Shadow shadow;
    Pooh(final String name,final State state){
        super(name,state);
        shadow = new Shadow("Тень "+name,State.STAY);
    }
    @Override
    public void enterInHouse(){
        System.out.println(name+" входит в дом");
    }
    public Shadow getShadow() {
        return shadow;
    }
    public void suggest(Entity entity) {
        getMind().setMood(Mood.HAPPY);
```

```
        System.out.println(name+" предлагает "+entity.name+" пойти в гости к Пятачку
после того как поест");
        System.out.println("Так как там, может быть, угостят желудями");
        getMind().setMood(Mood.NORMAL);
        setState(State.EATING,true,false);
    }
    public void tellAboutLittleCreature(Entity entity, LittleCreature littleCreature){
        System.out.println(name+" начинает рассказывать "+ entity +" о том, что
"+littleCreature.whoAmI());
        littleCreature.troubles();
        System.out.println(name+" просит "+entity+" не наскакивать во время первого
знакомства");
    }
    public void startJourneyWith(Entity entity, String place){
        setState(State.ON_WAY);
        entity.setState(State.ON_WAY);
        System.out.println(name+" и "+ entity +" после завтрака начали путешествие в
"+place);
    }
    public void endJourneyWith(Tigger tigger,Piglet piglet){
        System.out.println(name+" стучит в дверь");
        Guest[] guests = {this, tigger};
        piglet.openDoor(guests);
        setState(State.AT_HOUSE);
        tigger.setState(State.AT_HOUSE);
    }

    static class Shadow extends Entity {
        private boolean isCaught = false;
        Shadow(final String name,final State state){
            super(name,state);
        }
        public void turnAway(final Tigger tigger){
            if(!isCaught) {
                System.out.println(name + " отвернулась от " + tigger);
                tigger.tryToCatchShadow(this);
            }
        }
        public void setCaught(boolean caught) {
            isCaught = caught;
            if(isCaught){
                System.out.println(name+" была поймана!");
            }
            else{
                System.out.println(name+" не удалось поймать");
            }
        }
    }
}


// Piglet.java

package com.company;

public class Piglet extends Entity implements LittleCreature, HouseOwner{
    Piglet(final String name,final State state){
        super(name, state);
    }
    @Override
    public String whoAmI() {
        return name+" - очень маленькое существо";
    }
    @Override
    public void troubles(){
        System.out.println(name+" не любит, когда на него напрыгивают");
    }
    @Override
```

```
    public void giveFood(FoodTaker foodTaker){
        System.out.println(name+" подвинул корзинку с желудями к гостю");
        speak("Угощайтесь, пожалуйста");
        foodTaker.takeFood(new Nut());
    }
    @Override
    public void openDoor(Guest[] guests) throws NobodyAtParty{
        if(guests == null) throw new NobodyAtParty("Nobody goes to "+name);
        System.out.println(name+" открывает дверь");
        for(Guest i: guests){
            i.enterInHouse();
        }
    }
    public void hug(Pooh pooh){
        getMind().setMood(Mood.FEAR);
        System.out.println(name+" прижимается к "+pooh);
        getMind().setVoice(Voice.ALMOST_FUNNY);
    }
}
// Tigger.java

package com.company;

public class Tigger extends Entity implements FoodTaker, Guest{
    Tigger(final String name,final State state){
        super(name,state);
    }
    @Override
    public void takeFood(Item item){
        item.changeOwner(this);

    }
    @Override
    public void enterInHouse(){
        System.out.println(name+" входит в дом, прекращая прыгать");
    }
    public void playWithShadow(Pooh.Shadow shadow){
        System.out.println(name+" начинает прятаться за деревьями");
        shadow.turnAway(this);
        shadow.turnAway(this);
        System.out.println(name+" рассказывает, что Тигры наскакивают только до
завтрака");
        System.out.println("А как только они поедят желудей, то становятся Тихими и
Вежливыми");
    }
    public void sayAboutProblem(){
        getMind().setVoice(Voice.VERY_STRONG);
        speak("Поверь мне, ты хоть и очень гостеприимный, но я бы хотел желудей");
        getMind().setVoice(Voice.NORMAL);
    }
    public void tryToCatchShadow(Pooh.Shadow shadow){
        System.out.println(name + " пытается поймать " + shadow);
        shadow.setCaught(Math.random()<0.5);
    }
    public void tryToCall(Pooh pooh) throws PoohNotReadyException {
        System.out.println(name+" пытается отправится вместе с Пухом");
        if(pooh.getState() == State.EATING) throw new PoohNotReadyException();
        System.out.println(name+" рад, что они наконец-то отправятся в путь");
    }
    public void tryToFinishEating(){
        System.out.println(name+" долго пережевывает желуди");
        getMind().setVoice(Voice.UNCLEAR);
    }
}


// Voice.java
```

```java
package com.company;

public enum Voice {
    NORMAL("нормальный", false) {
        @Override
        public String getVoicePhrase() {
            return "говорит";
        }
    },
    ALMOST_FUNNY("почти весёлый", false) {
        @Override
        public String getVoicePhrase() {
            return "почти весёлым голосом произносит";
        }
    },
    CLOGGED_WITH_ACORNS("забитый желудями", true) {
        @Override
        public String getVoicePhrase() {
            return "не может говорить, так как его рот забит желудями";
        }
    },
    VERY_STRONG("очень уверенный",false) {
        @Override
        public String getVoicePhrase() { return "очень уверенно говорит"; }
    },
    UNCLEAR("непонятный",false) {
        @Override
        public String getVoicePhrase() { return "невнятно сказал"; }
    };

    public final String name;
    public final boolean isSilence;
    Voice(final String name,final boolean isSilence){
        this.name = name;
        this.isSilence = isSilence;
    }
    public abstract String getVoicePhrase();
    public String getName(){
        return name;
    }
}


// State.java

package com.company;

public enum State {
    STAY(null,null),
    EATING("начал кушать","закончил кушать"),
    ON_WAY("отправляется в путь","прибыл на место"),
    AT_HOUSE("теперь в доме","вышел из дома");
    private final String startPhrase;
    private final String endPhrase;
    State(final String startPhrase,final String endPhrase){
        this.startPhrase = startPhrase;
        this.endPhrase = endPhrase;
    }
    public String startState(){
        return startPhrase;
    }
    public String endState(){
        return endPhrase;
    }
}


// LittleCreature.java
```

```java
package com.company;

public interface LittleCreature {
    String whoAmI();
    void troubles();
}
```

```java
// HouseOwner.java

package com.company;

public interface HouseOwner {
    void giveFood(FoodTaker foodTaker);
    void openDoor(Guest[] guests);
}
```

```java
// Guest.java

package com.company;

public interface Guest {
    void enterInHouse();
}
```

```java
// FoodTaker.java

package com.company;

public interface FoodTaker {
    void takeFood(Item item);
}
```

```java
// NobodyAtParty.java

package com.company;

// unchecked
public class NobodyAtPartyException extends RuntimeException{
    NobodyAtParty(String msg){
        super(msg);
    }
}
```

```java
// PoohNotReadyException.java

package com.company;

// checked
public class PoohNotReadyException extends Exception {
    PoohNotReadyException(){
        super("!!! Пух еще кушает и не может отправиться в путь !!!");
    }
}
```

Результат работы

```
*голос у Тигра меняется на очень уверенный*
Тигра очень уверенно говорит: Поверь мне, ты хоть и очень гостеприимный, но я бы хотел желудей
*голос у Тигра меняется на нормальный*
*настроение у Пух меняется на довольный*
Пух предлагает Тигра пойти в гости к Пятачку после того как поест
Так как там, может быть, угостят желудями
*настроение у Пух меняется на нормальный*
*Пух начал кушать*

Тигра пытается отправится вместе с Пухом
!!! Пух еще кушает и не может отправиться в путь !!!
Тигра пытается отправится вместе с Пухом
!!! Пух еще кушает и не может отправиться в путь !!!
Пух психует и быстро доедает
*Пух закончил кушать*
Тигра пытается отправится вместе с Пухом
Тигра рад, что они наконец-то отправятся в путь

*Пух вышел из дома*
*Пух отправляется в путь*
*Тигра вышел из дома*
*Тигра отправляется в путь*
Пух и Тигра после завтрака начали путешествие в дом Пятачка

Пух начинает рассказывать Тигра о том, что Пятачок - очень маленькое существо
Пятачок не любит, когда на него напрыгивают
Пух просит Тигра не наскакивать во время первого знакомства
Тигра начинает прятаться за деревьями
Тень Пух отвернулась от Тигра
Тигра пытается поймать Тень Пух
Тень Пух не удалось поймать
Тень Пух отвернулась от Тигра
Тигра пытается поймать Тень Пух
Тень Пух была поймана!
Тигра рассказывает, что Тигры наскакивают только до завтрака
А как только они поедят желудей, то становятся Тихими и Вежливыми

Пух стучит в дверь
Пятачок открывает дверь
Пух входит в дом
Тигра входит в дом, прекращая прыгать
*Пух прибыл на место*
*Пух теперь в доме*
*Тигра прибыл на место*
*Тигра теперь в доме*

Пятачок подвинул корзинку с желудями к гостю
Пятачок говорит: Угощайтесь, пожалуйста
Тигра берёт орехи и начинает есть
*голос у Тигра меняется на забитый желудями*
*настроение у Пятачок меняется на испуганный*
Пятачок прижимается к Пух
*голос у Пятачок меняется на почти весёлый*
Пятачок почти весёлым голосом произносит: Так ты Тигра? Ну-ну!
Тигра не может говорить, так как его рот забит желудями
Тигра долго пережевывает желуди
*голос у Тигра меняется на непонятный*
Тигра невнятно сказал: Да, это я
Пух и Пятачок говорят: Что-что?
*голос у Тигра меняется на нормальный*
Тигра говорит: Я ЕСТЬ ТИГРА
```

Вывод

В ходе выполнения данной лабораторной работы я научился работ с исключениями Java, познакомился с внутренними и анонимными классами и применил их на практике. Обработка исключений и работа с классами является важной частью объектно-ориентированного программирования.