Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

**Лабораторная работа №3**

Вариант №32138.4

**Группа**: P3132

**Выполнил**: Русинов Дмитрий Станиславович

**Преподаватель:** Горбунов Михаил Витальевич

Санкт-Петербург

2021

Оглавление

[Текст задания 3](#_Toc88509894)

[UML диаграмма классов 4](#_Toc88509895)

[Исходный код программы 5](#_Toc88509896)

[Main 5](#_Toc88509897)

[Emotion 6](#_Toc88509898)

[AliveCreature 7](#_Toc88509899)

[Human 8](#_Toc88509900)

[IntelligentCreature 10](#_Toc88509901)

[MagicHat 11](#_Toc88509902)

[Well 12](#_Toc88509903)

[MarmeladeWell 12](#_Toc88509904)

[WaterWell 12](#_Toc88509905)

[Conversationable 13](#_Toc88509906)

[HasEmotions 13](#_Toc88509907)

[EmotionalEnfluenceable 13](#_Toc88509908)

[They 13](#_Toc88509909)

[Выходные данные программы 15](#_Toc88509910)

[Заключение 16](#_Toc88509911)

# Текст задания

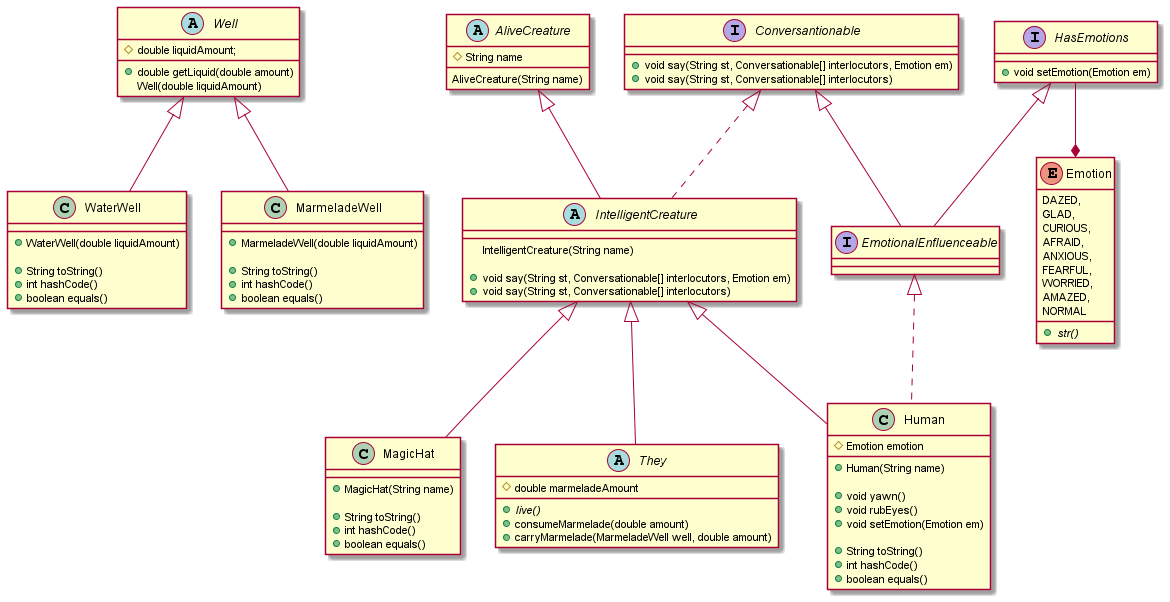
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

# UML диаграмма классов



# Исходный код программы

## Main

*import* abstractThings.Emotion;  
*import* creatures.Human;  
*import* creatures.MagicHat;  
*import* interfaces.Conversationable;  
  
*public class* Main {  
 *public static void* main(String[] args) {  
 Human alice = *new* Human("Alice");  
 Human sonya = *new* Human("Sonya");  
  
 MagicHat hat = *new* MagicHat("The Hat");  
  
  
 hat.say("Из обыкновенного колодца таскают воду," +  
 " а из мармеладного колодца всякий может, я надеюсь, таскать мармелад.", *new* Conversationable[]{alice, sonya});  
  
 alice.say("Ты что - совсем дурочка? " +  
 "- Я говорю, как они могли таскать мармелад оттуда? Ведь они там жили", *new* Conversationable[]{hat, sonya});  
  
 sonya.say("Не только жили! Они жили-были!", *new* Conversationable[]{hat, alice}, Emotion.DAZED);  
  
 System.out.println("\n");  
  
 sonya.yawn();  
 sonya.rubEyes();  
  
 System.out.println("\n");  
  
 sonya.say("Так вот, этот самый мармадад они ели и пили" +  
 " - делали что хотели...", *new* Conversationable[]{hat, alice});  
  
  
 }  
}

## Emotion

*package* abstractThings;  
  
*public enum* Emotion {  
 DAZED{  
 *@Override  
 public* String str() {  
 *return* "dazed";  
 }  
 },  
 GLAD{  
 *@Override  
 public* String str() {  
 *return* "glad";  
 }  
 },  
 CURIOUS{  
 *@Override  
 public* String str() {  
 *return* "curious";  
 }  
 },  
 AFRAID{  
 *@Override  
 public* String str() {  
 *return* "afraid";  
 }  
 },  
 ANXIOUS{  
 *@Override  
 public* String str() {  
 *return* "anxious";  
 }  
 },  
 FEARFUL{  
 *@Override  
 public* String str() {  
 *return* "fearful";  
 }  
 },  
 WORRIED{  
 *@Override  
 public* String str() {  
 *return* "worried";  
 }  
 },  
 AMAZED{  
 *@Override  
 public* String str() {  
 *return* "amazed";  
 }  
 },  
 NORMAL{  
 *@Override  
 public* String str() { *return* "normal"; }  
 };  
  
 *public abstract* String str();  
  
}

## AliveCreature

*package* creatures;  
  
*public abstract class* AliveCreature {  
 *protected* String name;  
 AliveCreature(String name){  
 *this*.name = name;  
 }  
}

## Human

*package* creatures;  
  
*import* abstractThings.Emotion;  
*import* interfaces.EmotionalEnfluenceable;  
  
*import* java.util.Objects;  
  
*public class* Human *extends* IntelligentCreature *implements* EmotionalEnfluenceable {  
  
 *public* Human(String name) {  
 *super*(name);  
 }  
  
 *protected* Emotion emotion = Emotion.NORMAL;  
  
 *public void* yawn() {  
 System.out.println(*this*.name + " yawns");  
 }  
  
 *public void* rubEyes() {  
 System.out.println(*this*.name + " rubs eyes");  
 }  
  
 *@Override  
 public void* setEmotion(Emotion em) {  
 *this*.emotion = em;  
 }  
  
 *public* Emotion getEmotion() {  
 *return this*.emotion;  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Human human = (Human) o;  
 *return* (emotion == human.emotion && name.equals(human.name));  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return* Objects.hash(emotion, name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "Human{" +  
 "name='" + name + '\'' +  
 ", emotion=" + emotion +  
 '}';  
 }  
}

## IntelligentCreature

*package* creatures;  
  
*import* abstractThings.Emotion;  
*import* interfaces.Conversationable;  
*import* interfaces.EmotionalEnfluenceable;  
  
*public abstract class* IntelligentCreature *extends* AliveCreature *implements* Conversationable {  
  
 IntelligentCreature(String name){  
 *super*(name);  
 }  
  
 *@Override  
 public void* say(String st, Conversationable[] interlocutors, Emotion em){  
 System.out.println(*this*.name + " said\": " + st + "\".");  
  
 *for* (Conversationable interlocutor: interlocutors){  
 *if* (interlocutor *instanceof* EmotionalEnfluenceable){  
  
 ((EmotionalEnfluenceable) interlocutor).setEmotion(em);  
 *if*(interlocutor *instanceof* AliveCreature){  
 System.out.println(((AliveCreature) interlocutor).name + " is now " + em.str() + " because of these words.");  
 }  
  
 *else*{  
 System.out.println("The thing is now" + em.str() + "because of these words.");  
 }  
 }  
 }  
  
 }  
  
 *@Override  
 public void* say(String st, Conversationable[] interlocutors) {  
 System.out.println(*this*.name + " said\": " + st + "\".");  
 }  
  
  
}

## MagicHat

*package* creatures;  
  
*import* java.util.Objects;  
  
*public class* MagicHat *extends* IntelligentCreature{  
 *public* MagicHat(String name){  
 *super*(name);  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return* Objects.hash(name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "MagicHat{" +  
 "name='" + name + '\'' +  
 '}';  
  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 MagicHat human = (MagicHat) o;  
 *return* name.equals(human.name);  
 }  
}

## Well

*package* things;  
  
*public abstract class* Well {  
 *protected double* liquidAmount;  
  
 Well(*double* liquidAmount) {  
 *this*.liquidAmount = liquidAmount;  
 }  
  
 *public double* getLiquid(*double* amount) {  
 *if* (liquidAmount - amount < 0) {  
 System.out.println("Can't use the Well because you want too much liquid.");  
 *return* 0;  
 } *else* {  
 liquidAmount -= amount;  
 System.out.println("The Well now has less liquid, got it successfully.");  
 *return* amount;  
 }  
 }  
}

## MarmeladeWell

*package* things;  
  
*public class* MarmeladeWell *extends* Well {  
 *public* MarmeladeWell(*double* liquidAmount) {  
 *super*(liquidAmount);  
 }  
}

## WaterWell

*package* things;  
  
*public class* WaterWell *extends* Well {  
 *public* WaterWell(*double* liquidAmount) {  
 *super*(liquidAmount);  
 }  
}

## Conversationable

*package* interfaces;  
  
*import* abstractThings.Emotion;  
  
*public interface* Conversationable {  
 *public void* say(String st, Conversationable[] interlocutors, Emotion em);  
 *public void* say(String st, Conversationable[] interlocutors);  
}

## HasEmotions

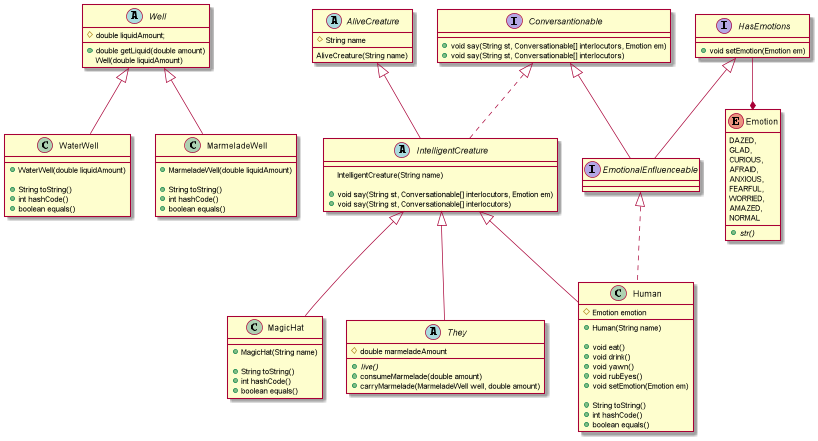
*package* interfaces;  
  
*import* abstractThings.Emotion;  
  
*public interface* HasEmotions {  
 *void* setEmotion(Emotion em);  
}

## EmotionalEnfluenceable

*package* interfaces;  
  
*public interface* EmotionalEnfluenceable *extends* Conversationable, HasEmotions {  
}

## They

*package* creatures;  
  
*import* things.MarmeladeWell;  
  
*public abstract class* They *extends* IntelligentCreature{  
 *protected double* marmeladeAmount = 0;  
  
 *public* They(){  
 *super*("they");  
 }  
  
 *public abstract void* live();  
  
 *public void* carryMarmelade(MarmeladeWell well, *double* amount){  
 *double* got = well.getLiquid(amount);  
 *this*.marmeladeAmount += got;  
 }  
  
 *public void* consumeMarmelade(*double* amount){  
 *if* (marmeladeAmount - amount < 0){  
 System.out.println(name + " wanted to consume more marmelade than they have");  
 } *else* {  
 System.out.println(name + "successfully consumed " + amount + " of Marmelade");  
 }  
 }  
}



# Выходные данные программы

Изображение выглядит как текст

Автоматически созданное описание

# Заключение

Для выполнения этой лабораторной работы необходимо было изучить принципы SOLID, научиться работать с интерфейсами и enum’ами.