Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант № 282498

*Выполнил:*

Студент группы P3132

Янковой Леонид

Кириллович

*Преподаватель:*

Письмак Алексей Евгеньевич



Санкт-Петербург, 2022

Содержание

[Описание предметной области 3](#_Toc120216464)

[Диаграмма классов 4](#_Toc120216465)

[Исходный код 5](#_Toc120216466)

[Main.java 5](#_Toc120216467)

[Character.java 5](#_Toc120216468)

[Eatable.java 6](#_Toc120216469)

[FoodContainer.java 6](#_Toc120216470)

[ReflectionType.java 6](#_Toc120216471)

[Drink.java 7](#_Toc120216472)

[Food.java 7](#_Toc120216473)

[Mirror.java 8](#_Toc120216474)

[Music.java 8](#_Toc120216475)

[Shorty.java 9](#_Toc120216476)

[Table.java 10](#_Toc120216477)

[Tray.java 11](#_Toc120216478)

[Waiter.java 12](#_Toc120216479)

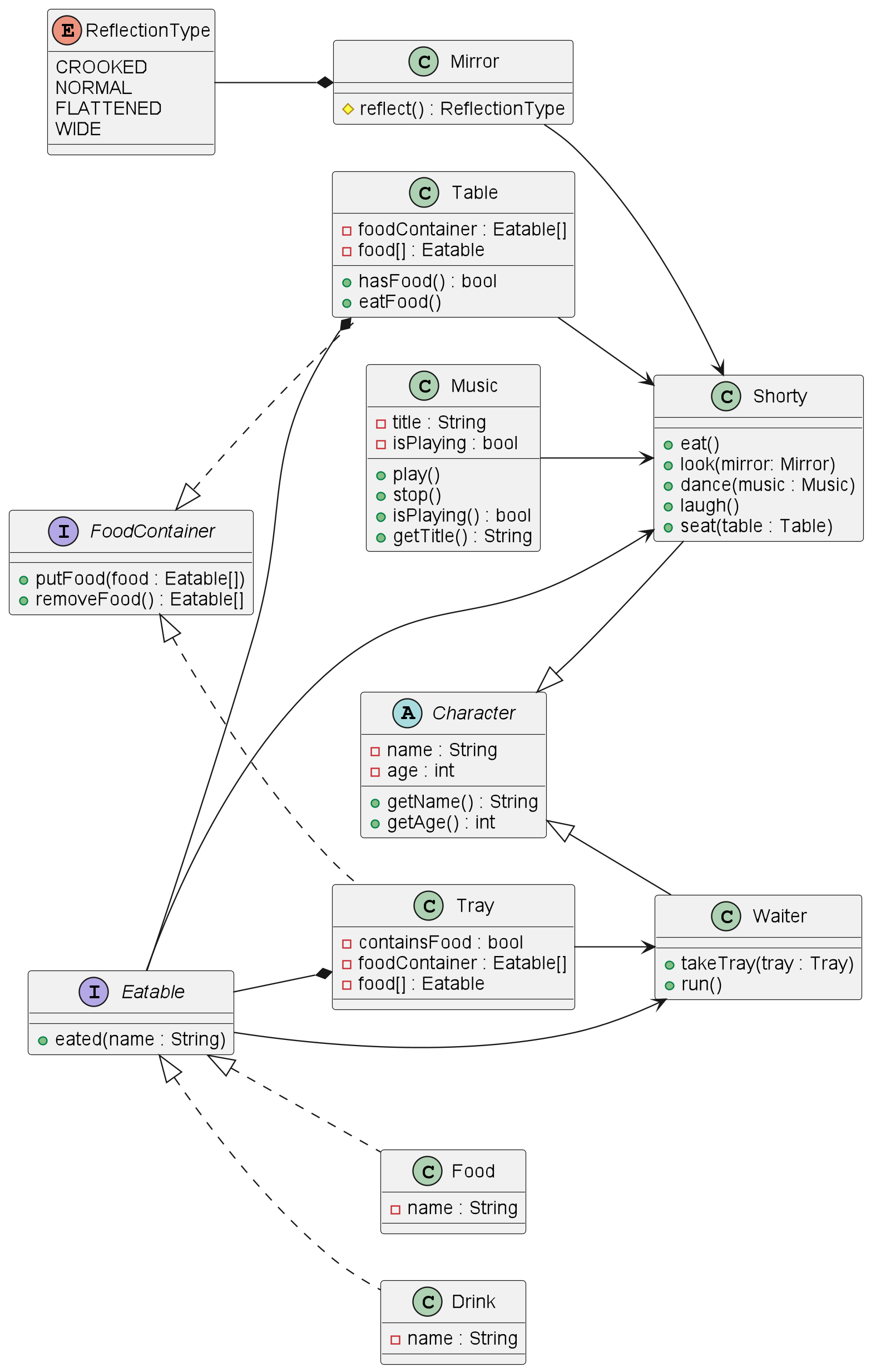
[Результат работы 13](#_Toc120216480)

[Вывод 14](#_Toc120216481)

# Описание предметной области

Внизу, вдоль тротуаров, были выставлены кривые зеркала, и каждый мог вдосталь нахохотаться, глядя на отражение своей вытянутой, сплюснутой или перекошенной самым неестественным образом физиономии. Тут же перед многочисленными столовыми и кафе, прямо на тротуаре, стояли столики. Многие коротышки сидели за столиками и ужинали, пили чай, кофе или газированную воду с сиропом, ели мороженое или просто закусывали. Некоторые танцевали тут же под музыку, которая гремела со всех сторон. Официанты и официантки бегали с подносами между столиками и приносили желающим разные кушанья.

# Диаграмма классов



# Исходный код

## Main.java

import core.\*;  
import utils.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Shorty pers1 = new Shorty("Пукич", 18);  
 Waiter waiter = new Waiter("Дмитрий Лянгузов", 18);  
 Mirror mirror = new Mirror();  
 Table table1 = new Table();  
 Tray tray1 = new Tray();  
 Tray tray2 = new Tray();  
 Music ostrov = new Music("0стр0в\_в0кеане");  
  
 pers1.look(mirror);  
 pers1.look(mirror);  
 pers1.seat(table1);  
 tray1.putFood(new Eatable[]{  
 new Food("Чезбарг"),  
 new Food("Картошечка"),  
 new Food("Ананас"),  
 new Drink("Квас"),  
 });  
 tray2.putFood(new Eatable[]{  
 new Food("Чезбарг"),  
 new Food("Картошечка"),  
 new Food("Ананас"),  
 new Drink("Квас"),  
 });  
 waiter.takeTray(tray1);  
 waiter.run();  
 waiter.unload(table1);  
 pers1.eat();  
 pers1.eat();  
 pers1.eat();  
 pers1.eat();  
 pers1.eat();  
  
 ostrov.play();  
 pers1.dance(ostrov);  
 ostrov.stop();  
  
 }  
}

## Character.java

package utils;  
  
import java.util.Objects;  
  
public abstract class Character {  
 protected String name;  
 protected int age;  
 public Character(String name, int age){  
 System.*out*.println("Персонаж "+ name+ " появляется");  
 this.age = age;  
 this.name = name;  
 }  
  
 public int getAge(){  
 return age;  
 }  
 public String toString(){  
 return name;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Character character = (Character) o;  
 return age == character.age && Objects.*equals*(name, character.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, age);  
 }  
}

## Eatable.java

package utils;  
  
public interface Eatable {  
 public void eated(String name);  
}

## FoodContainer.java

package utils;  
  
public interface FoodContainer {  
 public void putFood(Eatable[] food);  
 public Eatable[] removeFood();  
}

## ReflectionType.java

package utils;  
  
public enum ReflectionType{  
 *CROOKED*("скрюченное"),  
 *NORMAL*("обыкновенное"),  
 *FLATTENED*("сплющенное"),  
 *WIDE*("растянутое");  
  
 private String name;  
 private ReflectionType(String name){  
 this.name=name;  
 }  
  
 public String toString(){  
 return name;  
 }  
}

## Drink.java

package core;  
  
import utils.Eatable;  
  
import java.util.Objects;  
  
public class Drink implements Eatable {  
 private String name;  
 public Drink(String name){this.name = name;}  
  
 @Override  
 public void eated(String user) {  
 System.*out*.println(user+" выпивает "+name);  
 }  
  
 @Override  
 public String toString(){return name;}  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Drink drink = (Drink) o;  
 return Objects.*equals*(name, drink.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
}

## Food.java

package core;  
  
import utils.Eatable;  
  
import java.util.Objects;  
  
public class Food implements Eatable {  
 *// private int calories;* private String name;  
  
 public Food(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void eated(String user) {  
 System.*out*.println(user + " поедает " + name);  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Food food = (Food) o;  
 return name.equals(food.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
  
}

## Mirror.java

package core;  
  
import utils.ReflectionType;  
  
import java.util.Objects;  
import java.util.Random;  
  
  
public class Mirror {  
 ReflectionType[] reflections = new ReflectionType[]{ReflectionType.*WIDE*, ReflectionType.*CROOKED*, ReflectionType.*NORMAL*, ReflectionType.*FLATTENED*};  
 protected ReflectionType reflect() {  
 Random generator = new Random();  
 return reflections[generator.nextInt(4)];  
 }  
  
 @Override  
 public String toString() {  
 return "Зеркало";  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 return obj.getClass() == this.getClass();  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*("Mirror");  
 }  
}

## Music.java

package core;  
  
import java.util.Objects;  
  
public class Music {  
 private String title;  
 private boolean isPlaying;  
  
 public Music(String title){this.title = title;}  
 public void play(){isPlaying = true; System.*out*.println("Музыка "+title+" начинает играть...");}  
 public void stop(){isPlaying = false; System.*out*.println("Музыка "+title+" затухает...");}  
  
 public boolean isPlaying() {  
 return isPlaying;  
 }  
 @Override  
 public String toString(){return title;}  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Music music = (Music) o;  
 return title.equals(music.title);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title);  
 }  
}

## Shorty.java

package core;  
  
import utils.Character;  
import utils.ReflectionType;  
  
public class Shorty extends Character {  
 private Table table;  
  
*// private boolean sex; // sex == пол (ламинат, паркет итд)* public Shorty(String name, int age) {  
 super(name, age);  
*// this.sex = sex;* }  
  
 public void eat() {  
 if (!table.hasFood()) {  
 table.eatFood().eated(name);  
 } else {  
 System.*out*.println("Стол пуст, есть нечего!");  
 }  
 }  
  
 public void seat(Table table) {  
 System.*out*.println(name + " садится за стол");  
 this.table = table;  
 }  
  
 public void dance(Music music) {  
 if (music.isPlaying()) {  
 System.*out*.println(name + " танцует под музыку " + music.toString());  
 }  
 }  
  
 private void laugh() {  
 System.*out*.println(name + " смеется");  
 }  
  
 public void look(Mirror mirror) {  
 ReflectionType reflection = mirror.reflect();  
 System.*out*.println(name + " смотрит в зеркало и видит свое " + reflection.toString() + " отражение");  
 if (reflection == ReflectionType.*CROOKED*) {  
 laugh();  
 }  
 }  
  
}

## Table.java

package core;  
  
import utils.Eatable;  
import utils.FoodContainer;  
  
import java.util.EmptyStackException;  
import java.util.Objects;  
import java.util.Stack;  
  
public class Table implements FoodContainer {  
  
 *// private Eatable[] food = new Eatable[10];* private Stack<Eatable> foodContainer = new Stack<>();  
  
 public boolean hasFood(){  
 return foodContainer.isEmpty();  
 }  
  
 public Eatable eatFood() {  
 try {  
 return foodContainer.pop();  
 } catch (EmptyStackException e){  
 System.*out*.println("Стол пустой! кушать нечего!");  
 return null;  
 }  
 }  
  
 @Override  
 public void putFood(Eatable[] food) {  
 System.*out*.println("На стол ставят: ");  
 for (Eatable f : food) {  
 foodContainer.push(f);  
 System.*out*.println(f.toString());  
 }  
 }  
  
 @Override  
 public Eatable[] removeFood() {  
 System.*out*.println("Со стола убрали " + foodContainer.pop().toString());  
 Eatable[] empty = {};  
 return empty;  
 }  
 @Override  
 public String toString() {  
 return "Поднос";  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Table table = (Table) o;  
 return Objects.*equals*(foodContainer, table.foodContainer);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(foodContainer);  
 }  
}

## Tray.java

package core;  
  
import utils.Eatable;  
import utils.FoodContainer;  
  
*//import javax.lang.model.type.NullType;*import java.util.Arrays;  
import java.util.Objects;  
  
public class Tray implements FoodContainer {  
 private boolean containsFood = false;  
 private Eatable[] foodContainer;  
  
 @Override  
 public void putFood(Eatable[] food) {  
 foodContainer = food;  
 containsFood = true;  
 }  
  
 @Override  
 public Eatable[] removeFood() {  
 if (containsFood) {  
 return foodContainer;  
 } else {  
 System.*out*.println("Поднос пустой!");  
 Eatable[] empty = {};  
 return empty;  
 }  
 }  
 @Override  
 public String toString() {  
 return "Стол";  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Tray tray = (Tray) o;  
 return containsFood == tray.containsFood && Arrays.*equals*(foodContainer, tray.foodContainer);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = Objects.*hash*(containsFood);  
 result = 31 \* result + Arrays.*hashCode*(foodContainer);  
 return result;  
 }  
}

## Waiter.java

package core;  
  
import utils.Character;  
  
public class Waiter extends Character {  
 private Tray tray;  
 public Waiter(String name, int age){  
 super(name, age);  
 }  
 public void takeTray(Tray tray){  
 this.tray = tray;  
 System.*out*.println("Официант "+name+ " взял поднос");  
 }  
 public void run(){System.*out*.println("Официант "+name+ " бежит к посетителям");}  
 public void unload(Table table){  
 table.putFood(tray.removeFood());  
 }  
}

# Результат работы

Персонаж Паша появляется

Персонаж Дмитрий Лянгузов появляется

Паша смотрит в зеркало и видит свое обыкновенное отражение

Паша смотрит в зеркало и видит свое скрюченное отражение

Паша смеется

Паша садится за стол

Официант Дмитрий Лянгузов взял поднос

Официант Дмитрий Лянгузов бежит к посетителям

На стол ставят:

Чезбарг

Картошечка

Ананас

Квас

Паша выпивает Квас

Паша поедает Ананас

Паша поедает Картошечка

Паша поедает Чезбарг

Стол пуст, есть нечего!

Музыка 0стр0в\_в0кеане начинает играть...

Паша танцует под музыку 0стр0в\_в0кеане

Музыка 0стр0в\_в0кеане затухает...

# Вывод

Во время выполнения данной лабораторной работы я научился применять принципы SOLID на практике, изучил понятия абстрактного класса, интерфейса и перечисления, а так же столкнулся с основными плюсом и минусом SOLID: очень сложно начать проект без предварительной разработки архитектуры и представления необходимых зависимостей, однако гораздо легче расширять и дополнять готовый проект, в котором используются принципы SOLID.