Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Программирование»

**Отчет**

По лабораторной работе №3

Вариант 130130

Студент:

Казаев М. П.

*P3111*

Преподаватель:

Харитонова А. Е.

Санкт-Петербург, 2022 г.

**Текст задания:**

Но Кенга, и Кристофер Робин, и Пятачок-- все стояли вокруг Крошки Ру, уговаривая его принять рыбий жир. И Ру говорил: "Может, не надо?"-- а Кенга говорила: "Ну-ну, милый Ру, вспомни, что ты мне обещал". Тигра подошел поближе и наклонился над спинкой кресла Ру. И вдруг он высунул язык, послышалось громкое "буль-буль", и, подскочив от удивления, Кенга вскрикнула: "Ох!"-- и ухватила ложку как раз в ту секунду, когда она уже исчезала в пасти Тигры. Ложку она спасла, но рыбий жир исчез.

**Программа должна удовлетворять следующим требованиям:**

Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).

Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).

В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().

Программа должна содержать как минимум один перечисляемый тип (enum).

**Порядок выполнения работы:**

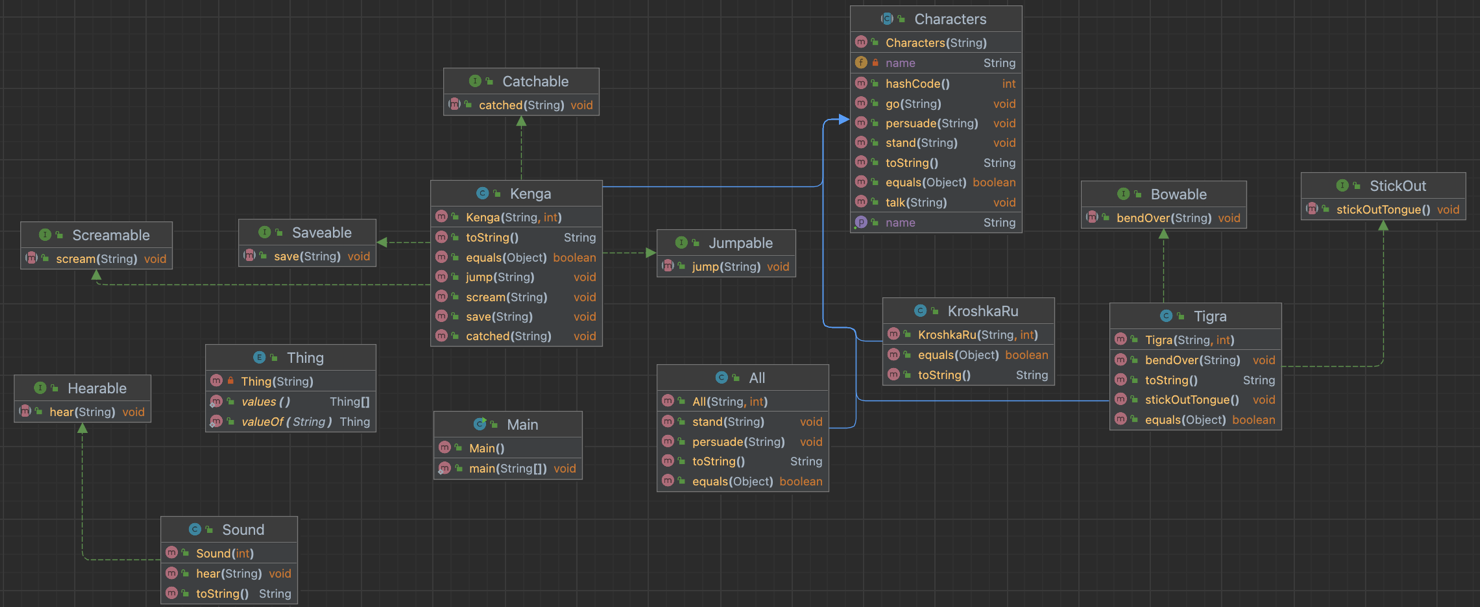
Доработать объектную модель приложения.

Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.

Согласовать с преподавателем изменения, внесённые в модель.

Модифицировать программу в соответствии с внесёнными в модель изменениями.

**UML-диаграмма классов**



**Код программы:**

public class All extends Characters {

private int cnt;

public All(String name, int cnt) {

super(name);

}

@Override

public void stand(String place) {

System.*out*.println("Персонажи " + super.getName() + " стояли " + place);

}

@Override

public void persuade(String things) {

System.*out*.println("Персонажи " + getName() + " уговаривали " + things);

}

@Override

public String toString() {

return "Кенга, Кристофер Робин и Пятачок" + cnt;

}

@Override

public boolean equals(Object obj) {

if (obj == this) {

return true;

}

if (obj == null || obj.getClass() != getClass()) {

return false;

}

All all = (All) obj;

return cnt == all.cnt;

}

}

public interface Bowable {

void bendOver(String place);

}

public interface Catchable {

void catched(String something);

}

public abstract class Characters {

private String name;

public String getName() {

return name;

}

public Characters(String name) {

this.name = name;

}

@Override

public int hashCode() {

return super.hashCode() + this.getName().hashCode();

}

@Override

public boolean equals(Object ob) {

if (ob == this) {

return true;

}

if (ob == null || ob.getClass() != getClass()) {

return false;

}

Characters ch = (Characters) ob;

return name == ch.name;

}

@Override

public String toString() {

return "Имя персонажа " + getName();

}

public void talk(String words) {

System.*out*.println("Персонаж " + name + " говорит: " + words);

}

public void go(String where) {

System.*out*.println("Персонаж " + name + " пошел " + where);

}

public void stand(String place) {

System.*out*.println("Персонаж " + name + " стоит " + place);

}

public void persuade(String things) {

System.*out*.println("Персонаж " + name + " угоравивает " + things);

}

}

public interface Hearable {

void hear(String sound);

}

public interface Jumpable {

void jump(String reason);

}

public class Kenga extends Characters implements Jumpable, Saveable, Screamable, Catchable {

private int age;

public Kenga(String name, int age) {

super(name);

}

@Override

public void catched(String something) {

System.*out*.println("Персонаж Кенга схватил " + something);

}

@Override

public void jump(String reason) {

System.*out*.println("Персонаж Кенга подскочил от " + reason);

}

@Override

public void save(String thing) {

System.*out*.println("Персонаж Кенга спас " + thing);

}

@Override

public void scream(String word) {

System.*out*.println("Персонаж Кенга вскрикнул " + word);

}

@Override

public String toString() {

return "Кенга" + age;

}

@Override

public boolean equals(Object obj) {

if (obj == this) {

return true;

}

if (obj == null || obj.getClass() != getClass()) {

return false;

}

Kenga kenga = (Kenga) obj;

return age == kenga.age;

}

}

public class KroshkaRu extends Characters {

private int age;

public KroshkaRu(String name, int age) {

super(name);

}

@Override

public String toString() {

return "Крошка РУ" + age;

}

@Override

public boolean equals(Object obj) {

if (obj == this) {

return true;

}

if (obj == null || obj.getClass() != getClass()) {

return false;

}

KroshkaRu kroshkaRu = (KroshkaRu) obj;

return age == kroshkaRu.age;

}

}

public interface Saveable {

void save(String thing);

}

public interface Screamable {

public void scream(String word);

}

public class Sound implements Hearable {

private int frequency;

public Sound(int frequency){}

@Override

public void hear(String sound) {

System.*out*.println("Послышался звук " + sound);

}

@Override

public String toString() {

return "звук" + frequency;

}

@Override

public boolean equals(Object obj) {

if (obj == this) {

return true;

}

if (obj == null || obj.getClass() != getClass()) {

return false;

}

Sound sound = (Sound) obj;

return frequency == sound.frequency;

}

}

public interface StickOut {

void stickOutTongue();

}

public enum Thing {

*SPOON*("Объект ложка исчез"),

*FISHFAT*("Объект рыбий жир исчез");

Thing(String s) {

System.*out*.println(s);

}

}

public class Tigra extends Characters implements Bowable, StickOut {

private int age;

public Tigra(String name, int age) {

super(name);

}

@Override

public void bendOver(String place) {

System.*out*.println("Персонаж Тигра наклонился над " + place);

}

@Override

public void stickOutTongue() {

System.*out*.println("Персонаж Тигра высунул язык");

}

@Override

public String toString() {

return "Тигра" + age;

}

@Override

public boolean equals(Object obj) {

if (obj == this) {

return true;

}

if (obj == null || obj.getClass() != getClass()) {

return false;

}

Tigra tigra = (Tigra) obj;

return age == tigra.age;

}

}

public class Main {

public static void main(String[] args) {

Kenga kenga = new Kenga("Кенга");

Tigra tigra = new Tigra("Тигра");

KroshkaRu krosh = new KroshkaRu("Крошка Ру");

Sound sound = new Sound();

All all = new All("Кенга, Кристофер Робин и Пятачок");

all.stand("вокруг");

all.persuade("принять рыбий жир");

krosh.talk("Может не надо");

kenga.talk("Ну-ну, милый Ру, вспомни, что ты мне обещал");

tigra.go("ближе");

tigra.bend\_over("над спинкой кресла");

tigra.stick\_out\_tongue();

sound.hear("буль-буль");

kenga.jump("удивления");

kenga.scream("Ox!");

kenga.\_catch("ложку");

Thing spoon = Thing.*spoon*;

Thing fishfat = Thing.*fishfat*;

spoon.disappear(spoon);

fishfat.disappear(fishfat);

System.*out*.println(kenga.hashCode());

}

}

**Результат работы программы:**

Персонажи Кенга, Кристофер Робин и Пятачок стояли вокруг

Персонажи Кенга, Кристофер Робин и Пятачок уговаривали принять рыбий жир

Персонаж Крошка Ру говорит: Может не надо

Персонаж Кенга говорит: Ну-ну, милый Ру, вспомни, что ты мне обещал

Персонаж Тигра пошел ближе

Персонаж Тигра наклонился над над спинкой кресла

Персонаж Тигра высунул язык

Послышался звук буль-буль

Персонаж Кенга подскочил от удивления

Персонаж Кенга вскрикнул Ox!

Персонаж Кенга схватил ложку

Объект ложка исчез

Объект рыбий жир исчез

**Вывод:**

Во время выполнения лабораторной работы я научился работать с абстрактными классами, интерфейсами и нумераторами. Лучше освоил работу с полями, методами и классами. Также я познакомился с принципами SOLID, и узнал методы класса Object.