Федеральное государственное автономное образовательное учреждение высшего

образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Отчет**

**По лабораторной работе №3**

По дисциплине «Программирование»

Вариант 210037.

Выполнил: Чураков Александр Алексеевич, группа P3131

Преподаватель: Письмак Алексей Евгеньевич, преподаватель факультета ПИиКТ.

Санкт-Петербург

~ 2023 ~

Оглавление

[Задание 3](#_Toc151664309)

[Диаграмма классов реализованной объектной модели 3](#_Toc151664310)

[Исходный код 3](#_Toc151664311)

[Main 3](#_Toc151664312)

[Перечисления 3](#_Toc151664313)

[Languages 3](#_Toc151664314)

[BodyParts 3](#_Toc151664315)

[FindTime 3](#_Toc151664316)

[Pots 3](#_Toc151664317)

[Stuff 3](#_Toc151664318)

[Персонажи 3](#_Toc151664319)

[Абстрактный персонаж 3](#_Toc151664320)

[Кристофер Робин 3](#_Toc151664321)

[Кенга 3](#_Toc151664322)

[Крошка Ру 3](#_Toc151664323)

[Пятачок 3](#_Toc151664324)

[Тигра 3](#_Toc151664325)

[Тигр 3](#_Toc151664326)

[Интерфейсы 3](#_Toc151664327)

[UgovarivAble 3](#_Toc151664328)

[AskAble 3](#_Toc151664329)

[FindAble 3](#_Toc151664330)

[SpeakAble 3](#_Toc151664331)

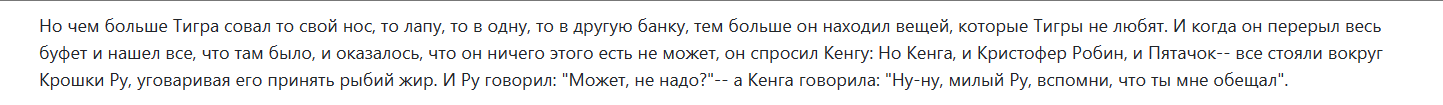
[StickAble 3](#_Toc151664332)

[Результат работы 3](#_Toc151664333)

[Вывод 3](#_Toc151664334)

# Задание

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



«Но чем больше Тигра совал то свой нос, то лапу, то в одну, то в другую банку, тем больше он находил вещей, которые Тигры не любят. И когда он перерыл весь буфет и нашел все, что там было, и оказалось, что он ничего этого есть не может, он спросил Кенгу: 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".»

**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать [принципам SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)).
2. Программа должна содержать как минимум два интерфейса и один абстрактный класс (номенклатура должна быть согласована с преподавателем).
3. В разработанных классах должны быть переопределены методы equals(), toString() и hashCode().
4. Программа должна содержать как минимум один перечисляемый тип (enum).

**Порядок выполнения работы:**

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.

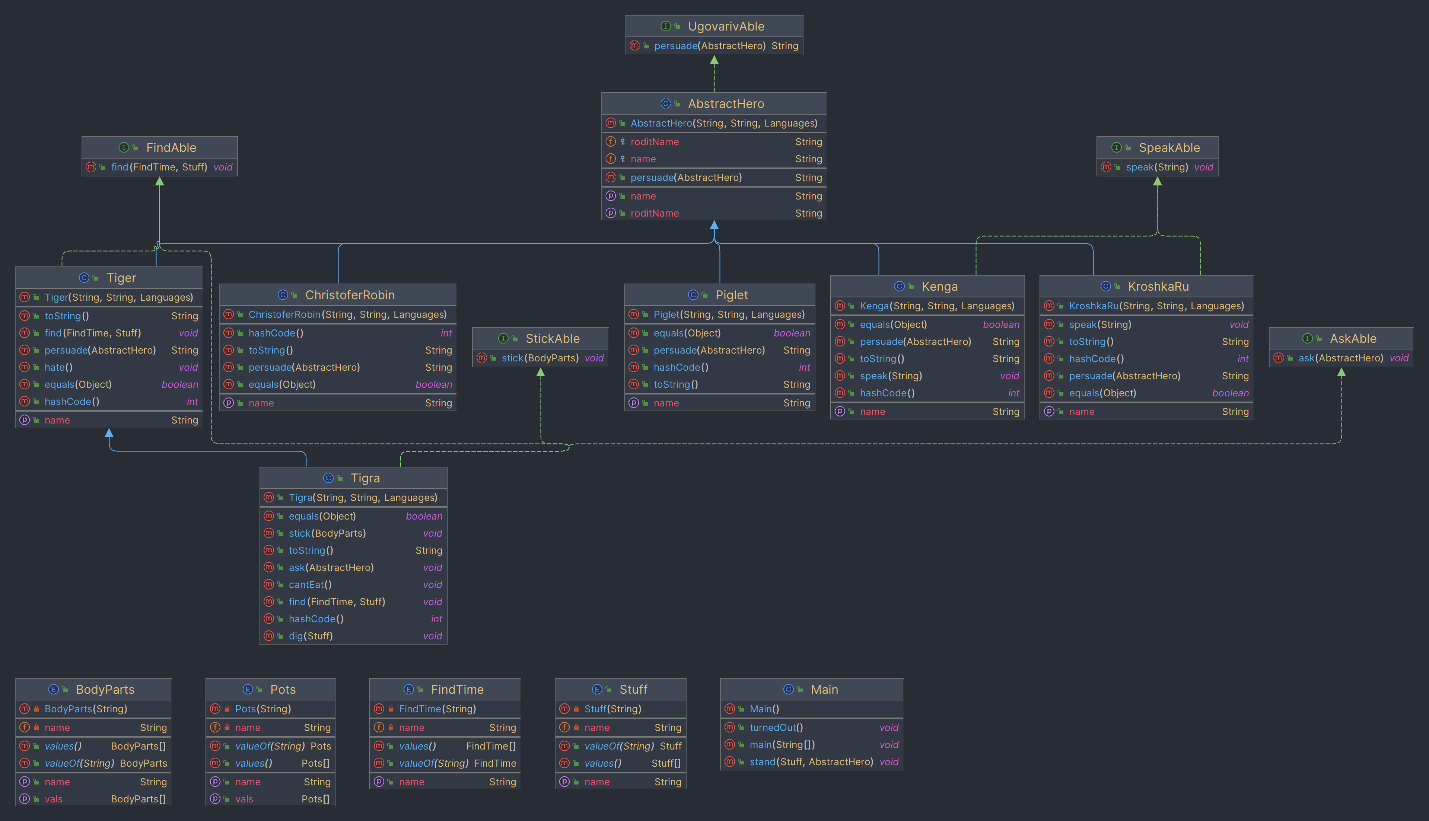
**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов объектной модели.
3. Исходный код программы.
4. Результат работы программы.
5. Выводы по работе.

**Вопросы к защите лабораторной работы:**

1. Принципы объектно-ориентированного программирования SOLID и STUPID.
2. Класс Object. Реализация его методов по умолчанию.
3. Особенности реализации наследования в Java. Простое и множественное наследование.
4. Понятие абстрактного класса. Модификатор abstract.
5. Понятие интерфейса. Реализация интерфейсов в Java, методы по умолчанию. Отличия от абстрактных классов.
6. Перечисляемый тип данных (enum) в Java. Особенности реализации и использования.
7. Методы и поля с модификаторами static и final.
8. Перегрузка и переопределение методов. Коварианты возвращаемых типов данных.
9. Элементы функционального программирования в синтаксисе Java. Функциональные интерфейсы, лямбда-выражения. Ссылки на методы.

# Диаграмма классов реализованной объектной модели



# Исходный код

Весь проект - <https://github.com/awesoma31/ITMO_Labs/tree/main/PROG/lab3>

## Main

*import* src.enums.BodyParts;  
*import* src.enums.FindTime;  
*import* src.enums.Languages;  
*import* src.enums.Stuff;  
*import* src.heroes.\*;  
*import* src.heroes.abstractClasses.AbstractHero;  
*import* src.interfaces.UgovarivAble;  
  
*public class* Main {  
 *public static void* turnedOut() {  
 System.out.print(" и оказалось, что ничего этого ");  
 }  
  
 *public static void* stand(Stuff stuff, AbstractHero hero) {  
 System.out.print(  
 stuff.getName() + " стояли вокруг " + hero.getRoditName()  
 );  
 }  
  
 *public static void* main(String[] args) {  
 Tigra tigra = *new* Tigra("Тигра", "Тигры", Languages.TIGERLANG);  
 Tiger tiger = *new* Tiger("Тигры", "Тигров", Languages.TIGERLANG);  
 ChristoferRobin christoferRobin = *new* ChristoferRobin(  
 "Кристофер Робин", "Кристофера Робина", Languages.HUMANLANG  
 );  
 Kenga kenga = *new* Kenga("Кенга", "Кенгу", Languages.KANGAROOOLANG);  
 KroshkaRu kroshkaRu = *new* KroshkaRu("Крошка Ру", "Крошки Ру", Languages.KANGAROOOLANG);  
 Piglet piglet = *new* Piglet("Пятачок", "Пятачка", Languages.PIGLETLANG);  
  
 UgovarivAble[] whoPersuaded = {kenga, christoferRobin, piglet};  
  
 tigra.stick(BodyParts.NOSE);  
  
 tigra.find(FindTime.PAST, Stuff.THINGS);  
 tiger.hate();  
 tigra.dig(Stuff.ALL);  
 System.out.println();  
 tiger.find(FindTime.PRESENT, Stuff.ALL);  
  
 turnedOut();  
  
 tigra.cantEat();  
 tigra.ask(kenga);  
  
 System.out.println();  
 stand(Stuff.ALL, kroshkaRu);  
 System.out.println();  
  
 *for* (UgovarivAble persuadingHero : whoPersuaded) {  
 System.out.println(persuadingHero.persuade(kroshkaRu));  
 }  
  
 kroshkaRu.speak("Может, не надо?");  
 System.out.println();  
 kenga.speak("Ну-ну, милый Ру, вспомни, что ты мне обещал");  
 }  
}

## Перечисления

### Languages

*package* src.enums;  
  
*public enum* Languages {  
 TIGERLANG("тигрином"),   
 HUMANLANG("человеческом"),   
 PIGLETLANG("пятачковском"),   
 KANGAROOOLANG("кенгурушном");  
  
 *private final* String title;  
  
 Languages(String s) {  
 title = s;  
 }  
  
 *public* String getTitle() {  
 *return* title;  
 }  
}

### BodyParts

*package* src.enums;  
  
*public enum* BodyParts {  
 NOSE("нос"), PAW("лапу");  
  
 *private final* String name;  
  
 BodyParts(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public* BodyParts[] getVals() {  
 *return* values();  
 }  
}

### FindTime

*package* src.enums;  
  
*public enum* FindTime {  
 PAST("находил"), PRESENT("нашел");  
  
 *private final* String name;  
  
 FindTime(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
}

### Pots

*package* src.enums;  
  
*public enum* Pots {  
 ONE("одну"),  
 OTHER("другую");  
  
 *private final* String name;  
  
 Pots(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public* Pots[] getVals() {  
 *return* Pots.values();  
 }  
}

### Stuff

*package* src.enums;  
  
*public enum* Stuff {  
 THINGS("вещей"), ALL("все");  
  
 *private final* String name;  
  
 Stuff(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
}

## Персонажи

### Абстрактный персонаж

*package* src.heroes.abstractClasses;  
  
  
*import* src.enums.Languages;  
*import* src.interfaces.UgovarivAble;  
  
*public abstract class* AbstractHero *implements* UgovarivAble {  
 *protected* String name;  
 *protected* String roditName;  
 *protected* Languages language;  
  
 *public* AbstractHero(String name, String roditName, Languages lang) {  
 *this*.name = name;  
 *this*.roditName = roditName;  
 language = lang;  
 }  
  
 *public abstract* String persuade(AbstractHero who);  
  
 *public abstract* String getName();  
  
 *public* String getRoditName() {  
 *return* roditName;  
 }  
}

### Кристофер Робин

*package* src.heroes;  
  
*import* src.enums.Languages;  
*import* src.heroes.abstractClasses.AbstractHero;  
  
*import* java.util.Objects;  
  
*public class* ChristoferRobin *extends* AbstractHero {  
 *public* ChristoferRobin(String name, String roditName, Languages lang) {  
 *super*(name, roditName, lang);  
 }  
  
 *@Override  
 public* String getName() {  
 *return* name;  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return super*.hashCode() + *this*.getName().hashCode();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 ChristoferRobin that = (ChristoferRobin) o;  
 *return* Objects.equals(name, that.name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "ChristoferRobin{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return this*.name + " уговаривал " + who.getRoditName() + " на " + language.getTitle() + " языке";  
 }  
}

### Кенга

*package* src.heroes;  
  
*import* src.enums.Languages;  
*import* src.heroes.abstractClasses.AbstractHero;  
*import* src.interfaces.SpeakAble;  
  
*import* java.util.Objects;  
  
*public class* Kenga *extends* AbstractHero *implements* SpeakAble {  
 *public* Kenga(String name, String roditName, Languages lang) {  
 *super*(name, roditName, lang);  
 }  
  
 *@Override  
 public* String getName() {  
 *return this*.name;  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return super*.hashCode() + *this*.getName().hashCode();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Kenga that = (Kenga) o;  
 *return* Objects.equals(name, that.name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "Кенгу";  
 }  
  
 *@Override  
 public void* speak(String str) {  
 System.out.print(" - а Кенга говорит: \"" + str + "\".");  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return this*.name + " уговаривал " + who.getRoditName() + " на " + language.getTitle() + " языке";  
 }  
}

### Крошка Ру

*package* src.heroes;  
  
*import* src.enums.Languages;  
*import* src.heroes.abstractClasses.AbstractHero;  
*import* src.interfaces.SpeakAble;  
  
*import* java.util.Objects;  
  
*public class* KroshkaRu *extends* AbstractHero *implements* SpeakAble {  
 *public* KroshkaRu(String name, String roditName, Languages lang) {  
 *super*(name, roditName, lang);  
 }  
  
 *@Override  
 public* String getName() {  
 *return this*.name;  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return super*.hashCode() + *this*.getName().hashCode();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 KroshkaRu that = (KroshkaRu) o;  
 *return* Objects.equals(name, that.name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "KroshkaRu{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return this*.name + " уговаривал " + who.getRoditName() + " на " + language.getTitle() + " языке";  
 }  
  
 *@Override  
 public void* speak(String str) {  
 System.out.print(" И " + *this*.name + " говорит: \"" + str + "\"");  
 }  
}

### Пятачок

*package* src.heroes;  
  
*import* src.enums.Languages;  
*import* src.heroes.abstractClasses.AbstractHero;  
  
*import* java.util.Objects;  
  
*public class* Piglet *extends* AbstractHero {  
  
 *public* Piglet(String name, String roditName, Languages lang) {  
 *super*(name, roditName, lang);  
 }  
  
  
 *@Override  
 public* String getName() {  
 *return* name;  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return super*.hashCode() + *this*.getName().hashCode();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Piglet that = (Piglet) o;  
 *return* Objects.equals(name, that.name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "Piglet{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return this*.name + " уговаривал " + who.getRoditName() + " на " + language.getTitle() + " языке";  
 }  
}

### Тигра

*package* src.heroes;  
  
*import* src.enums.\*;  
*import* src.heroes.abstractClasses.AbstractHero;  
*import* src.interfaces.AskAble;  
*import* src.interfaces.FindAble;  
*import* src.interfaces.StickAble;  
  
*import* java.util.Objects;  
  
*public class* Tigra *extends* Tiger *implements* StickAble, FindAble, AskAble {  
 *public* Tigra(String name, String roditName, Languages lang) {  
 *super*(name, roditName, lang);  
 }  
  
 *@Override  
 public void* find(FindTime time, Stuff stuff) {  
 System.out.print(" тем больше " + *this*.name + " " + time.getName() + " " + stuff.getName());  
 }  
  
 *@Override  
 public void* stick(BodyParts bp) {  
 System.out.print("Но чем больше " + *this*.name + " совал свой ");  
 *for* (BodyParts v : bp.getVals()) {  
 System.out.print(v.getName() + " ");  
 }  
  
 *for* (Pots pt : Pots.getVals()) {  
 System.out.print(" то в ");  
 System.out.print(pt.getName());  
 }  
 System.out.print(" банку");  
 }  
  
 *public void* cantEat() {  
 System.out.print("есть не может, ");  
 }  
  
 *@Override  
 public void* ask(AbstractHero obj) {  
 System.out.print(" он спросил " + obj.getRoditName() + ": ");  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return super*.hashCode() + *this*.getName().hashCode();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Tigra that = (Tigra) o;  
 *return* Objects.equals(name, that.name);  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "Tigra{" + name + "}";  
 }  
  
 *public void* dig(Stuff stuff) {  
 System.out.print(" И когда он перерыл " + stuff.getName() + " весь буфет,");  
 }  
}

### Тигр

*package* src.heroes;  
  
*import* src.enums.FindTime;  
*import* src.enums.Languages;  
*import* src.enums.Stuff;  
*import* src.heroes.abstractClasses.AbstractHero;  
*import* src.interfaces.FindAble;  
  
*import* java.util.Objects;  
  
*public class* Tiger *extends* AbstractHero *implements* FindAble {  
 *public* Tiger(String name, String roditName, Languages lang) {  
 *super*(name, roditName, lang);  
 }  
  
 *@Override  
 public* String getName() {  
 *return* name;  
 }  
  
 *public void* hate() {  
 System.out.print(" которые " + *this*.name + " не любят.");  
 }  
  
 *@Override  
 public void* find(FindTime time, Stuff stuff) {  
 System.out.print(time.getName() + " " + stuff.getName() + " что там было");  
 }  
  
 *@Override  
 public int* hashCode() {  
 *return super*.hashCode() + *this*.getName().hashCode();  
 }  
  
 *@Override  
 public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (o == *null* || getClass() != o.getClass()) *return false*;  
 Tiger that = (Tiger) o;  
 *return* Objects.equals(name, that.name);  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return this*.name + " уговаривал " + who.getRoditName() + " на " + language.getTitle() + " языке";  
 }  
  
 *@Override  
 public* String toString() {  
 *return* "Tiger{" +  
 "name='" + name + '\'' +  
 ", roditName='" + roditName + '\'' +  
 '}';  
 }  
}

## Интерфейсы

### UgovarivAble

*package* src.interfaces;  
  
*import* src.heroes.abstractClasses.AbstractHero;  
  
*public interface* UgovarivAble {  
 String persuade(AbstractHero who);  
}

### AskAble

*package* src.interfaces;  
  
*import* src.heroes.abstractClasses.AbstractHero;  
  
*public interface* AskAble {  
 *void* ask(AbstractHero whom);  
}

### FindAble

*package* src.interfaces;  
  
*import* src.enums.FindTime;  
*import* src.enums.Stuff;  
  
*public interface* FindAble {  
 *void* find(FindTime time, Stuff stuff);  
}

### SpeakAble

*package* src.interfaces;  
  
*public interface* SpeakAble {  
 *void* speak(String whatToSay);  
}

### StickAble

*package* src.interfaces;  
  
*import* src.enums.BodyParts;  
*import* src.enums.Pots;  
  
*public interface* StickAble {  
 *void* stick(BodyParts obj);  
}

# Результат работы

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

# Вывод

Во время выполнения работы я ознакомился с принципами SOLID, научился составлять модель проекта и составлять UML диаграммы.