Федеральное государственное автономное образовательное учреждение высшего

образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Отчет**

**По лабораторной работе №4**

По дисциплине «Программирование»

Вариант 210037,5.

Выполнил: Чураков Александр Алексеевич, группа P3131

Преподаватель: Письмак Алексей Евгеньевич, преподаватель факультета ПИиКТ.

Санкт-Петербург

~ 2023 ~

Оглавление

[Задание 4](#_Toc154012774)

[Диаграмма классов реализованной объектной модели 5](#_Toc154012775)

[Исходный код 5](#_Toc154012776)

[Main 5](#_Toc154012777)

[Перечисления 5](#_Toc154012778)

[Moods 5](#_Toc154012779)

[Languages 5](#_Toc154012780)

[BodyParts 5](#_Toc154012781)

[FindTime 5](#_Toc154012782)

[Pots 5](#_Toc154012783)

[Stuff 5](#_Toc154012784)

[Volume 5](#_Toc154012785)

[Exceptions 5](#_Toc154012786)

[Checked 5](#_Toc154012787)

[Unchecked 5](#_Toc154012788)

[Персонажи 5](#_Toc154012789)

[Абстрактный персонаж 5](#_Toc154012790)

[Кристофер Робин 5](#_Toc154012791)

[Кенга 5](#_Toc154012792)

[Крошка Ру 5](#_Toc154012793)

[Пятачок 5](#_Toc154012794)

[Пух 5](#_Toc154012795)

[Тигра 5](#_Toc154012796)

[Тигр 5](#_Toc154012797)

[Story 5](#_Toc154012798)

[Интерфейсы 5](#_Toc154012799)

[DisappearAble 5](#_Toc154012800)

[UgovarivAble 5](#_Toc154012801)

[AskAble 5](#_Toc154012802)

[JumpAble 5](#_Toc154012803)

[FindAble 5](#_Toc154012804)

[SpeakAble 5](#_Toc154012805)

[ThinkAble 5](#_Toc154012806)

[Understandable 5](#_Toc154012807)

[StickAble 5](#_Toc154012808)

[Результат работы 6](#_Toc154012809)

[Вывод 6](#_Toc154012810)

# Задание

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

Крошка Ру сказал «Здравствуй, Пух», и «Здравствуй, Пятачок» (по одному разу), и «Здравствуй, Тигра»  
(два раза, потому что это очень забавно звучало и, кроме того, он ведь никогда еще так не здоровался), они рассказали Кенге,  
зачем они пришли, и Кенга очень ласково сказала: «Ну что ж, милый Тигра, загляни в мой буфет и посмотри — что тебе там понравится».  
Ведь Кенга сразу поняла, что, хотя с виду Тигра очень большой, он так же нуждается в ласке, как и Крошка Ру.  
— А можно мне тоже поглядеть? — сказал Пух который уже начал себя чувствовать немножко одиннадцатичасно, и, получив согласие,  
он разыскал небольшую банку сгущенного молока. Что-то, видимо, подсказало ему, что Тигры не любят сгущенного молока, и  
он тихонечко унес банку в уголок и спокойно занялся ею.  
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:  
— Что же теперь будет?  
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».  
— Что это там такое? — шепнул Тигра Пятачку.  
— Это ему лекарство дают, — сказал Пятачок. — Витамины! Он их ненавидит!  
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:**

1. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
2. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

**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. Обработка исключительных ситуаций, три типа исключений.
2. Вложенные, локальные и анонимные классы.
3. Механизм рефлексии (reflection) в Java. Класс Class.

# Диаграмма классов реализованной объектной модели

Изображение выглядит как текст, снимок экрана, диаграмма

Автоматически созданное описание

# Исходный код

Весь проект - <https://github.com/awesoma31/ITMO_Labs/tree/main/PROG/lab4>

## Main

*import* enums.\*;  
*import* exceptions.checked.PasswdNotFoundException;  
*import* exceptions.unchecked.UnableToContinueStoryException;  
*import* heroes.\*;  
*import* heroes.abstractClasses.AbstractHero;  
*import* interfaces.DisappearAble;  
*import* interfaces.UgovarivAble;  
*import* story.Story;  
  
*import* java.util.Scanner;  
  
*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) {  
 Scanner in = *new* Scanner(System.in);  
   
 Runtime.getRuntime().addShutdownHook(*new* Thread(){  
 *public void* run() {  
 System.out.println("prog is dead");  
 }  
 });  
  
 Story story = *new* Story();  
 Story.Password.check("docs/passwd");  
  
 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);  
 Puh puh = *new* Puh("Пух", "Пуха", Languages.HUMANLANG);  
  
 kroshkaRu.say("Здравствуй Пух");  
 kroshkaRu.say("Здравствуй Пятачок");  
 kroshkaRu.tell(kenga, " зачем они пришли ");  
  
 System.out.println("Введите `1`, чтобы продолжить историю или `0`, чтобы остановить");  
 Story.Valve valve = story.*new* Valve();  
 story.continueStory(in.nextInt(), valve);  
  
 kenga.say(tigra, "Ну что ж, милый Тигра, загляни в мой буфет и посмотри — что тебе там понравится");  
 kenga.understand("хотя с виду Тигра очень большой, он так же нуждается в ласке, как и Крошка Ру");  
  
 puh.say("А можно мне тоже поглядеть?");  
 puh.setMood(Moods.ELEV\_OCLOCK);  
 puh.find(Stuff.POT);  
 puh.think("Тигры не любят сгущенного молока");  
 puh.eat(Stuff.POT);  
  
 */////////////* 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");  
 *////////////////* tigra.say("Что это там такое?", piglet, Volume.QUIETLY);  
 piglet.say(tigra, "Это ему лекарство дают. — Витамины! Он их ненавидит!");  
  
 tigra.bend(Stuff.CHAIR\_BACK);  
 tigra.stickOut(BodyParts.TOUNGUE);  
  
 System.out.println("послышался буьл-буль");  
  
 kenga.jump(" от удивления");  
 kenga.say("Ох!", kenga, Volume.LOUD);  
 kenga.catchSpoon(*true*);  
  
 DisappearAble fishOil = *new* DisappearAble() {  
 *@Override  
 public void* disappear() {  
 String name = "ложка";  
 System.out.println("но " + name + " исчезла");  
 }  
 };  
 fishOil.disappear();  
 }  
}

## Перечисления

### Moods

*package* enums;  
  
*public enum* Moods {  
 ELEV\_OCLOCK("одинадцатичасно");  
  
 Moods(String s) {  
 }  
}

### 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* enums;  
  
*public enum* BodyParts {  
 NOSE("нос"), PAW("лапу"), TOUNGUE("язык");  
  
 *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* enums;  
  
*public enum* Stuff {  
 THINGS("вещей"), ALL("все"), POT("банка"), CHAIR\_BACK("спинка кресла Ру");  
  
 *private final* String name;  
  
 Stuff(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
}

### Volume

*package* enums;  
  
*public enum* Volume {  
  
 LOUD("громко"),  
 QUIETLY("тихо");  
  
 Volume(String s) {  
 }  
}

## Exceptions

### Checked

*package* exceptions.checked;  
  
*import* java.io.FileNotFoundException;  
  
*public class* PasswdNotFoundException *extends* FileNotFoundException {  
 *public* PasswdNotFoundException(String message) {  
 *super*(message);  
 }  
}

### Unchecked

*package* exceptions.unchecked;  
  
*public class* UnableToContinueStoryException *extends* RuntimeException{  
 *public* UnableToContinueStoryException(String msg) {  
 *super*(msg);  
 }  
}

## Персонажи

### Абстрактный персонаж

*package* heroes.abstractClasses;  
  
  
*import* enums.Languages;  
*import* 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* AbstractHero() {  
 *super*();  
 }  
  
 *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* heroes;  
  
*import* enums.Languages;  
*import* enums.Volume;  
*import* heroes.abstractClasses.AbstractHero;  
*import* interfaces.JumpAble;  
*import* interfaces.SpeakAble;  
*import* interfaces.UnderstandAble;  
  
*import* java.util.Objects;  
  
*public class* Kenga *extends* AbstractHero *implements* SpeakAble, UnderstandAble, JumpAble {  
 *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 void* say(String s, AbstractHero hero, Volume volume) {  
 System.out.println(name + " сказала: " + s + volume.name());  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return this*.name + " уговаривал " + who.getRoditName() + " на " + language.getTitle() + " языке";  
 }  
  
 *@Override  
 public void* understand(String s) {  
 System.out.println(name + " понял " + s);  
 }  
  
 *@Override  
 public void* jump(String s) {  
 System.out.println("подскочила от " + s);  
 }  
  
 *public void* catchSpoon(*boolean* status) {  
 *final boolean* isCatched = status;  
  
 *class* Spoon {  
 *public void* printStatus() {  
 *if* (isCatched) {  
 System.out.println(name + " схватила ложку");  
 } *else* {  
 System.out.println(name + "не схватила ложку");  
 }  
  
 }  
 }  
  
 Spoon spoon = *new* Spoon();  
 spoon.printStatus();  
 }  
}

### Крошка Ру

*package* heroes;  
  
*import* enums.Languages;  
*import* heroes.abstractClasses.AbstractHero;  
*import* 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 void* say(String str) {  
 System.out.println(*this*.name + " сказал " + "\"" + str + "\"");  
 }  
  
 *@Override  
 public void* tell(AbstractHero whom, String s) {  
 System.out.println(*this*.name + " рассказал " + whom.getName() + ": " + s);  
 }  
  
 *@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* heroes;  
  
*import* enums.Languages;  
*import* heroes.abstractClasses.AbstractHero;  
*import* interfaces.SpeakAble;  
  
*import* java.util.Objects;  
  
*public class* Piglet *extends* AbstractHero *implements* SpeakAble {  
  
 *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";  
 }  
  
 *@Override  
 public void* say(AbstractHero hero, String str) {  
 System.out.println(name + " сказал " + hero.getName() + str);  
 }  
}

### Пух

*package* heroes;  
  
*import* enums.FindTime;  
*import* enums.Languages;  
*import* enums.Moods;  
*import* enums.Stuff;  
*import* heroes.abstractClasses.AbstractHero;  
*import* interfaces.FindAble;  
*import* interfaces.SpeakAble;  
*import* interfaces.ThinkAble;  
  
*public class* Puh *extends* AbstractHero *implements* SpeakAble, FindAble, ThinkAble {  
 *protected* Moods mood;  
  
 *public void* setMood(Moods mood) {  
 *this*.mood = mood;  
 }  
  
 *public* Puh (String name, String rn, Languages languages) {  
 *super*(name, rn, languages);  
 }  
  
 *@Override  
 public* String persuade(AbstractHero who) {  
 *return null*;  
 }  
  
 *@Override  
 public* String getName() {  
 *return null*;  
 }  
  
 *@Override  
 public void* speak(String whatToSay) {  
 System.out.println(name + " сказал " + whatToSay);  
 }  
  
 *@Override  
 public void* say(AbstractHero hero, String str) {  
 SpeakAble.*super*.say(hero, str);  
 }  
  
 *@Override  
 public void* find(Stuff stuff) {  
 System.out.println(name + " нашел " + stuff.getName());  
 }  
  
 *@Override  
 public void* think(String s) {  
 System.out.println(name + " думает: " + s);  
 }  
  
 *public void* eat(Stuff stuff) {  
 System.out.println(name + " ест " + stuff.getName());  
 }  
}

### Тигра

*package* heroes;  
  
*import* enums.\*;  
*import* heroes.abstractClasses.AbstractHero;  
*import* interfaces.AskAble;  
*import* interfaces.FindAble;  
*import* interfaces.SpeakAble;  
*import* interfaces.StickAble;  
  
*import* java.util.Objects;  
  
*public class* Tigra *extends* Tiger *implements* StickAble, FindAble, AskAble, SpeakAble {  
 *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,");  
 }  
  
 *@Override  
 public void* say(String s, AbstractHero hero, Volume volume) {  
 System.out.println(name + " сказад " + s + hero.getName() + volume);  
 }  
  
 *public void* bend(Stuff stuff) {  
 System.out.println(name + " наклонился над " + stuff.getName());  
 }  
  
 *public void* stickOut(BodyParts bp) {  
 System.out.println(name + " вытащил " + bp.getName());  
 }  
}

### 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 + '\'' +  
 '}';  
 }  
}

## Story

*package* story;  
  
*import* exceptions.unchecked.UnableToContinueStoryException;  
  
*import* java.util.Objects;  
  
*public class* Story {  
 *public class* Valve {  
 *private int* isOpen;  
  
 *public void* open() {  
 isOpen = 1;  
 }  
 }  
 *public static class* Password {  
 *public static void* check(String path){  
 *if* (!Objects.equals(path, "docs/passwd")) {  
 *throw new* UnableToContinueStoryException("Пароль неверен или не найден");  
 }  
 }  
 }  
  
 *public void* continueStory(*int* s, Story.Valve valve) *throws* UnableToContinueStoryException {  
 *if* (s == 1) {  
 valve.open();  
 } *else* {  
 *throw new* UnableToContinueStoryException("История остановлена");  
 }  
 }  
}

## Интерфейсы

### DisappearAble

*package* interfaces;  
  
*public interface* DisappearAble {  
 *void* disappear();  
}

### 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);  
}

### JumpAble

*package* interfaces;  
  
*public interface* JumpAble {  
 *void* jump(String s);  
}

### FindAble

*package* interfaces;  
  
*import* enums.FindTime;  
*import* enums.Stuff;  
  
*public interface* FindAble {  
 *default void* find(FindTime time, Stuff stuff) {  
 System.out.println("нашел " + stuff + time);  
 }  
 *default void* find(Stuff stuff) {  
 System.out.println("нашел " + stuff);  
 }  
}

### SpeakAble

*package* interfaces;  
  
*import* enums.Volume;  
*import* heroes.abstractClasses.AbstractHero;  
  
*public interface* SpeakAble {  
 *default void* speak(String whatToSay) {  
 System.out.println(whatToSay);  
 }  
  
 *default void* say(AbstractHero hero, String str) {  
 System.out.println(hero.getName() + " сказал " + str);  
 }  
  
 *default void* say(String str) {  
 System.out.println("сказал " + str);  
 }  
  
 *default void* tell(AbstractHero whom, String s) {  
 System.out.println("рассказал " + whom + ": " + s);  
 }  
  
 *default void* say(String s, AbstractHero hero, Volume volume) {  
 System.out.println("сказал " + s + hero.getName() + volume);  
 }  
}

### ThinkAble

*package* interfaces;  
  
*public interface* ThinkAble {  
 *default void* think(String s) {  
 System.out.println("думает: " + s);  
 }  
}

### Understandable

*package* interfaces;  
  
*import* heroes.abstractClasses.AbstractHero;  
  
*public interface* UnderstandAble {  
 *default void* understand(AbstractHero hero, String s) {  
 System.out.println(hero.getName() + " понял " + s);  
 }  
 *default void* understand(String s) {  
 System.out.println("понял " + s);  
 }  
}

### StickAble

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

# Результат работы

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

# Вывод

Во время выполнения работы я ознакомился с обработкой исключений, разными типами классов.