Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Программирование»

**Отчет**

По лабораторной работе №4

Вариант 158877.9

Студент

*Собитов Анвархон Акмалович*

*P3115*

Преподаватель

*Письмак Алексей Евгеньевич*

Санкт-Петербург, 2023г.

Оглавление

[Текст задания: 3](#_Toc125121367)

[Код программы: 4](#_Toc125121368)

[Результат работы программы 8](#_Toc125121369)

[Вывод: 9](#_Toc125121370)

# Текст задания:

И как только он сказал "легко", ветка, на которой он стоял, внезапно сломалась, и он, чувствуя, что падает, едва-едва успел вцепиться в верхнюю ветку. Затем он медленно, медленно поднял голову, зацепился подбородком за эту ветку... Затем подтянул одну заднюю лапу... Потом другую... И, наконец он уселся на эту ветку, тяжело дыша и от души жалея, что он вместо всего этого не попробовал заняться плаванием. Ру слез с Тигры и уселся рядышком. Ру немного помолчал, а потом спросил: Тигра сказал: Так они и сделали. Тем временем Винни-Пух и Пятачок продолжали свою прогулку. Пух в стихах сообщал Пятачку, что "неважно, чем он занят, так как он толстеть не станет, а ведь он толстеть не станет"; а Пятачок размышлял о том, скоро ли вырастет посаженный им желудь. Пятачок схватил Пуха за лапу, очевидно, чтобы Пух не очень пугался. Пух кивнул. Однако Пятачка это почему-то не утешило. Он все-таки чувствовал, что не стоит подходить к дереву, с которого, того гляди, кто-то бросится, хотя бы и очень умело, и он уже собирался побежать домой по какому-то очень срочному делу, когда Ягуляр подал голос. В ответ донесся чей-то восторженный писк с той самой ветки, где сидел Ягуляр:

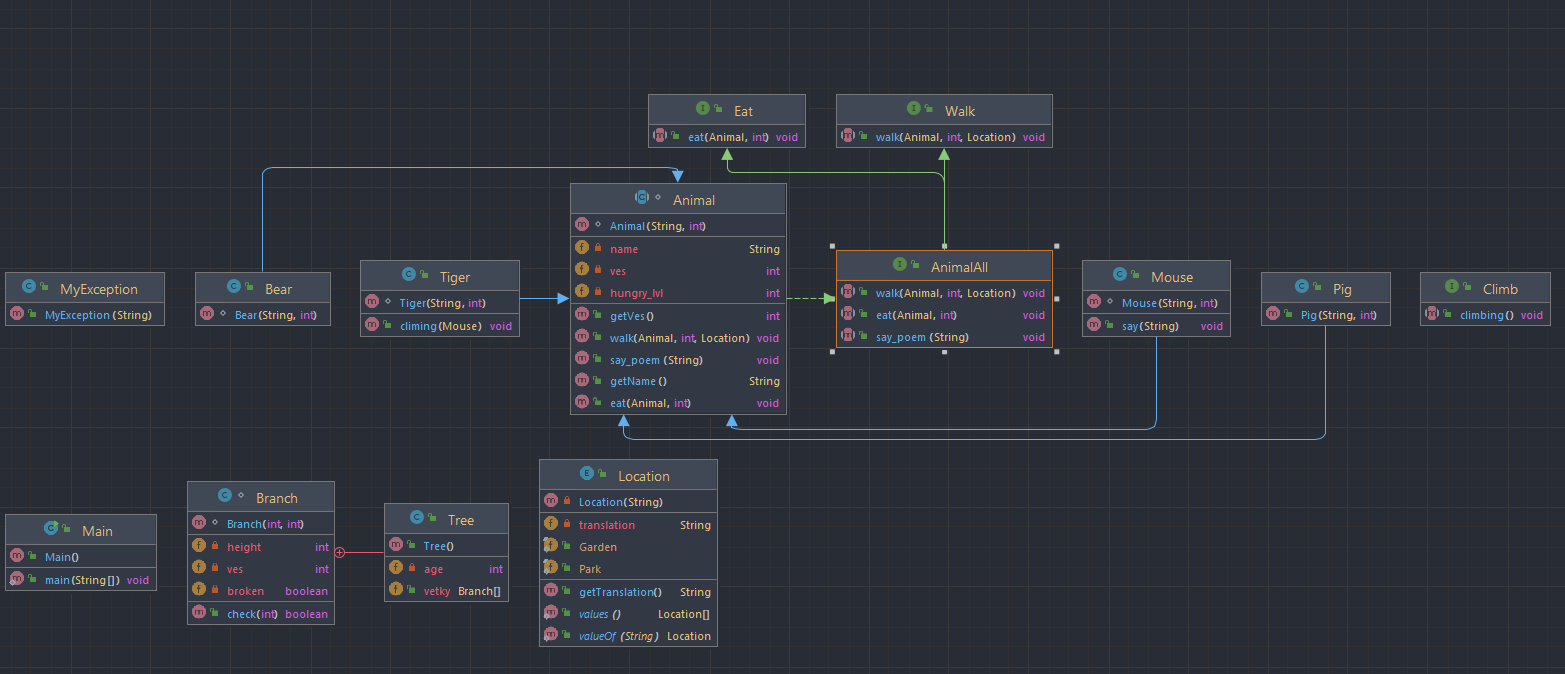
Программа должна удовлетворять следующим требованиям:

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.*

UML-диаграмма классов



# Код программы:

public class Main {  
 public static void main(String[] args) throws MyException {  
 Bear vini = new Bear("Вини-Пух", 10);  
 Pig ptchok = new Pig("Пятачок", 8);  
 Tiger tiger = new Tiger("Ягуляр", 21);  
 Mouse mouse = new Mouse("Ру", 5);  
  
  
 tiger.climing(mouse);  
 try {  
 vini.walk(ptchok, 5, Location.Park);  
 }catch (Exception e){  
 System.out.println("Они слышком устали Укажите значение km меньше десяти");  
 }  
 ptchok.walk(vini, 2, Location.Garden);  
 }  
}

abstract class Animal implements AnimalAll {  
 private String name;  
 private int ves;  
 private int hungry\_lvl = 0;  
  
  
 Animal(String name, int ves) throws MyException {  
 this.name = name;  
 if (ves != 0) {  
 this.ves = ves;  
 }else {  
 throw new MyException("Вес должен быть больше нуля");  
 }  
 }  
 public String getName() {  
 return name;  
 }  
  
 public int getVes() {  
 return ves;  
 }  
  
 @Override  
 public void walk(Animal animal, int km, Location location) throws MyException {  
 if(location == Location.Park) {  
 System.out.println(this.name + " прогулял с " + animal.getName() + "ом в " + Location.Park.getTranslation() );  
 if (km <= 10){  
 hungry\_lvl += km \* 10;  
 if(hungry\_lvl > 30) {  
 eat(animal, 40);  
 }  
 }else {  
 throw new MyException("Они так далеко не можеть идти.\n Установити значение km меньше десяти");  
 }  
 } else if(location == Location.Garden) {  
 System.out.println(this.name + " прогулял с " + animal.getName() + "ом в " + Location.Garden.getTranslation() );  
 System.out.println(this.name + " размышлял о том, скоро ли вырастет посаженный им желудь" );  
 }  
 }  
 @Override  
 public void eat(Animal animal, int kkl) {  
 System.out.println(this.name + " обедает с " + animal.getName()+"ом" +"\n");  
 say\_poem("Неважно, чем он занят, \n так как он толстеть не станет, \n а ведь он толстеть не станет \n ");  
 hungry\_lvl -= kkl;  
 }  
 @Override  
 public void say\_poem(String poem){  
 System.out.println(this.name + " рассказал стих:\n " + poem);  
 }  
  
}

public class Tiger extends Animal{  
 Tiger(String name, int ves) throws MyException {  
 super(name, ves);  
 }  
 public void climing(Mouse mouse){  
 Climb climb = new Climb() {  
 public int height = 0;  
 Tree tree = new Tree();  
 @Override  
 public void climbing() {  
 mouse.say("Можеш поднятся на дерево?");  
 for (int i = 0; i < tree.vetky.length; i++) {  
 System.out.println(Tiger.super.getName() + " поднялся на высату " + height);  
 if(tree.vetky[i].check(Tiger.super.getVes())) {  
 height += 5;  
 mouse.say("А ещё сможешь?");  
 }  
 else {  
 height -= 5;  
 System.out.println(Tiger.super.getName() + " едва-едва успел вцепиться в нижную ветку");  
 break;  
 }  
 }  
 height += 5;  
 }  
 };  
 climb.climbing();  
 }  
}

public class Tree {  
 private int age;  
 class Branch{  
 private int ves;  
 private int height;  
 private boolean broken = false;  
 Branch(int ves, int height) {  
 this.ves = ves;  
 this.height = height;  
 }  
 public boolean check(int ves) {  
 if(!broken) {  
 if(ves > this.ves) {  
 broken = true;  
 System.out.println("Ветка сломалась!");  
 return false;  
 }  
 else {  
 return true;  
 }  
 }  
 else return false;  
 }  
 }  
 public Branch[] vetky = {new Branch(30, 5), new Branch(30, 10), new Branch(20, 15)};  
}

public class Bear extends Animal{  
 Bear(String name, int ves) throws MyException {  
 super(name, ves);  
 }  
}

public class Mouse extends Animal{  
 Mouse(String name, int ves) throws MyException {  
 super(name, ves);  
 }  
  
 public void say(String text){  
 System.out.println(super.getName() + " сказал: " + text);  
 }  
}

public class Pig extends Animal{  
 public Pig(String name, int ves) throws MyException {  
 super(name, ves);  
 }  
}

public interface AnimalAll extends Eat, Walk{  
 void walk(Animal animal, int km, Location location) throws Exception;  
 void eat(Animal animal, int kkl);  
 void say\_poem(String poem);  
}

public interface Climb {  
 public void climbing();  
}

public interface Eat {  
 void eat(Animal animal, int kkl);  
}

public interface Walk {  
 void walk(Animal animal, int km, Location location) throws Exception ;  
}

public enum Location {  
 Park("Парк"),  
 Garden("Лес");  
 private String translation;  
  
 Location(String translation){  
 this.translation = translation;  
 }  
  
 public String getTranslation() {  
 return translation;  
 }  
}

public class MyException extends Exception{  
 public MyException(String msg){  
 super(msg);  
 }  
}

# Результат работы программы

Ру сказал: Можеш поднятся на дерево?

Ягуляр поднялся на высату 0

Ру сказал: А ещё сможешь?

Ягуляр поднялся на высату 5

Ру сказал: А ещё сможешь?

Ягуляр поднялся на высату 10

Ветка сломалась!

Ягуляр едва-едва успел вцепиться в нижную ветку

Вини-Пух прогулял с Пятачоком в Парк

Вини-Пух обедает с Пятачоком

Вини-Пух рассказал стих:

Неважно, чем он занят,

так как он толстеть не станет,

а ведь он толстеть не станет

Пятачок прогулял с Вини-Пухом в Лес

Пятачок размышлял о том, скоро ли вырастет посаженный им желудь

Process finished with exit code 0

# Вывод:

Во время выполнения лабораторной работы я научился работать с обработкой ошибок и исключений. Изучил локальные, абстрактные и внутренние классы. Лучше освоил работу с полями, методами и классами и построение UML диаграмм.