Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

ПИиКТ

Лабораторная работа №3

по дисциплине

«Программирование»

Вариант - 10565

Выполнил: Трубников Андрей P3110

Санкт-Петербург

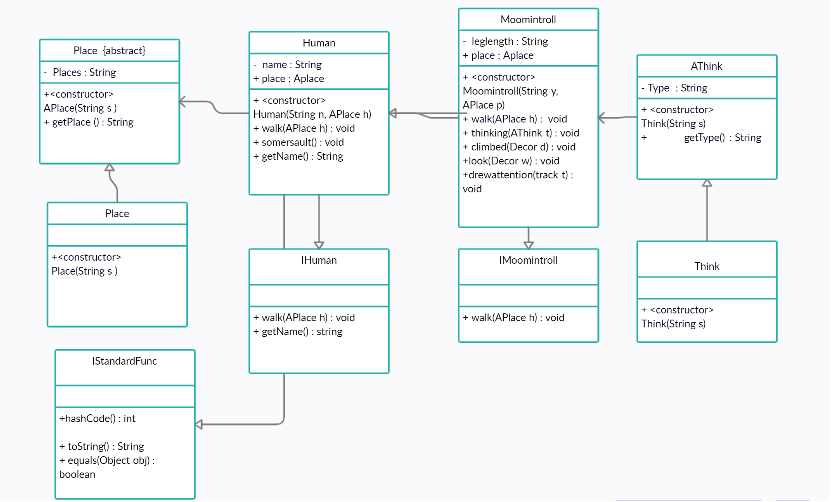
2020 г.

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

Он подошел к раскрытому окну и выглянул во двор. Ага, Снусмумрик воспользовался веревочной лестницей. Муми-тролль перебрался через подоконник и, осторожно переступая коротенькими ножками, спустился по лестнице вниз. На сырой земле отчетливо виднелись отпечатки ног Снусмумрика. Они были запутанные, словно куриный след, и не было никакой возможности определить, куда он направился. Местами следы делали длинные прыжки и перекрещивались между собой. "Это он от радости, -- размышлял про себя Муми-тролль. -- А вот тут он перекувырнулся, уж это точно".

**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).



**fairytale.java**

import javax.sound.midi.Track;  
  
public class fairytale {  
 public static void main(String[] args) {  
 System.*out*.println();  
 Place House = new Place("Дом");  
 Place courtyard = new Place("Двор");  
  
  
 Moomintroll Moom = new Moomintroll("25", House);  
 Think Per = new Think("Ага, Снусмумрик воспользовался веревочной лестницей.");  
 Human snusnu = new Human("Снусмумрик", House);  
 Decor window = new Decor();  
 Decor Stairs = new Decor();  
 Think kuv = new Think("Это он от радости. А вот тут он перекувырнулся, уж это точно.");  
 track sled = new track();  
  
 snusnu.walk(courtyard);  
 snusnu.somersault();  
 Moom.look(window);  
 Moom.thinking(Per);  
 Moom.climbed(Stairs);  
 Moom.walk(courtyard);  
 Moom.drewattention(sled);  
 Moom.thinking(kuv);  
 }  
}

**Moomintroll.java**

public class Moomintroll extends Human {  
 private String leglength;  
 APlace place;  
  
 Moomintroll(String y, APlace p) {  
 super(y, p);  
 }  
 public void thinking(AThink t){  
  
 System.*out*.println("Муми-тролль подумал :" + t.getType());  
 }  
 public void walk(APlace h){  
 place = h;  
 System.*out*.println("Муми-тролль пошагал на место: " + place.getPlace());  
 }  
 public void climbed(Decor d){  
 System.*out*.println("Муми-тролль спустился по :" + d.getStairs());  
 }  
 public void look(Decor w){  
  
 System.*out*.println("Муми-тролль выглянул в :" + w.getWindow());  
 }  
 public void drewattention(track t){  
 System.*out*.println("Муми-тролль увидел " + t.getTracks() + " Следы" );  
 }  
  
}

**IMoomintroll.java**

public interface IMoomintroll {  
 void walk(APlace h);  
 String getName();  
}

**Human.java**

public class Human implements IHuman, IStandardFunc {  
 private String Name;  
 APlace places;  
 Human(String n, APlace h) {  
 Name = n;  
 places = h;  
 }  
 public void walk(APlace h){  
 places = h;  
 System.*out*.println(Name + " пошагал на место: " + places.getPlace());  
 }  
  
 @Override  
 public String getName() {  
 return null;  
 }  
  
 @Override  
 public int hashCode() {  
 return super.hashCode()+Name.hashCode();  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 boolean comp = obj.hashCode()== this.hashCode() ? true : false;  
 return comp;  
 }  
 public void somersault(){  
 System.*out*.println(Name + " Делает кувырок.");  
 }  
}

**IHuman.java**

public interface IHuman {  
 void walk(APlace p);  
 String getName();  
}

**APlace.java**

abstract class APlace {  
 private String PlaceS;  
 APlace(String s) {  
 PlaceS = s;  
 }  
 String getPlace() {  
 return PlaceS;  
 }  
}

**Place.java**

public class Place extends APlace {  
 Place(String s) {  
 super(s);  
 }  
}

IStandardFunc.java

public interface IStandardFunc {  
 public int hashCode();  
 public String toString();  
 public boolean equals(Object obj);  
}

AThink.java

abstract class AThink {  
 private String Type;  
 AThink(String s) {  
 Type = s;  
 }  
 String getType() {  
 return Type;  
 }  
}

Think.java

public class Think extends AThink{  
 Think(String s) {  
 super(s);  
 }  
}

Décor.java

public class Decor {  
 private String Stairs;  
 public String getStairs() {  
 return " Лестница";  
 }  
 private String Window;  
 public String getWindow() {  
 return " Окно";  
 }  
 }

track.java

public class track {  
 private String Tracks ;  
  
 public enum kinds {  
 *Запутанные*,  
 *Куриные* ;  
 }  
 public String getTracks() {  
 kinds[] all = kinds.*values*();  
 Tracks = "";  
 for (kinds kinds : all) {  
 Tracks += kinds + " ";  
 }  
 return Tracks;  
 }  
}

Вывод программы :

Снусмумрик пошагал на место: Двор

Снусмумрик Делает кувырок.

Муми-тролль выглянул в : Окно

Муми-тролль подумал :Ага, Снусмумрик воспользовался веревочной лестницей.

Муми-тролль спустился по : Лестница

Муми-тролль пошагал на место: Двор

Муми-тролль увидел Запутанные Куриные Следы

Муми-тролль подумал :Это он от радости. А вот тут он перекувырнулся, уж это точно.

**Выводы:** Выполнив эту лабораторную работу, я изучил UML диаграммы и научился их читать. Теперь я лучше стал понимать различия между агрегацией, композицией и ассоциацией, наследованием и обращением, а также узнал об объектах, классах их полях и методах, способы их представления на диаграммах UML.