**Кафедра** **вычислительной** **техники**

**ОТЧЁТ** **ПО** **ЛАБОРАТОРНОЙ** **РАБОТЕ** **№3** «Программирование»

Вариант №3412285

Преподаватель: Письмак Алексей Евгеньевич

Выполнила: Иванова Вероника Вадимовна

гр. R3142

Санкт-Петербург 2020

**Задание:**

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

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.

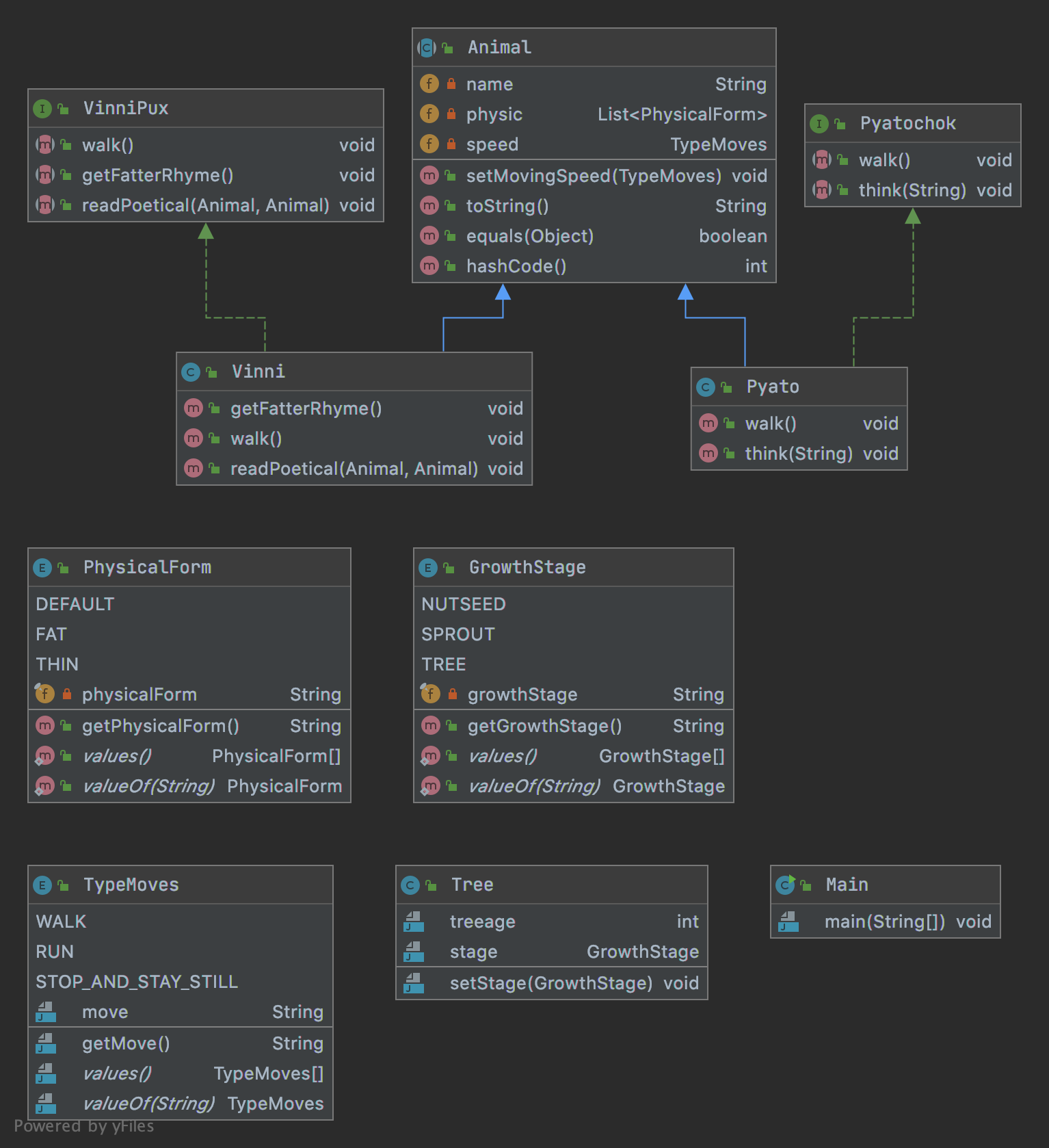
**Программа должна удовлетворять следующим требованиям:**

1. Доработанная модель должна соответствовать принципам SOLID.
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.

**UML диаграмма:**

****

**Код программы:**

public class Main {  
 public static void main(String[] args) {  
 Vinni vinni = new Vinni("Винни-Пух");  
 Pyato pyatochok = new Pyato("Пяточок");  
  
 vinni.setMovingSpeed(TypeMoves.*WALK*);  
 pyatochok.setMovingSpeed(TypeMoves.*WALK*);  
  
 vinni.walk();  
 pyatochok.walk();  
  
 vinni.readPoetical(vinni,pyatochok);  
 vinni.say("неважно, чем я занят, так как я толстеть не стану, а ведь я толстеть не стану");  
  
 pyatochok.think();  
 pyatochok.say("скоро "+ GrowthStage.*NUTSEED*.getGrowthStage() +" вырастет?");  
 }  
}

import java.util.List;  
import java.util.Objects;  
  
public abstract class Animal {  
  
 private String name;  
 private List<PhysicalForm> physic;  
 private TypeMoves speed;  
  
 public Animal(String name) {  
 this.name = name;  
  
 }  
  
 public void setMovingSpeed(TypeMoves speed) {  
 this.speed = speed;  
 }

@Override  
 public String toString() {  
 return this.name;  
 }  
  
 @Override  
 public boolean equals(Object ex) {  
 if (this == ex) return true;  
 if (this.getClass() != ex.getClass() || ex == null) return false;  
 Animal other = (Animal) ex;  
 if (name == other.name) return false;  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 return Math.*abs*(Objects.*hash*(this.name));  
 }  
  
}

public interface VinniPux {  
 void walk();  
 void getFatter();  
 void readPoetical(Animal teller, Animal receiver);  
}

public class Vinni extends Animal implements VinniPux {  
 public Vinni(String name) {  
 super(name);  
 }

@Override  
public void getFatterRhyme() {  
 System.*out*.println("\"неважно, чем я занят, так как я толстеть не стану, а ведь я толстеть не стану\"");  
}  
  
@Override  
public void walk() {  
 System.*out*.println(this + " " + TypeMoves.*WALK*.getMove());

}  
  
@Override  
public void readPoetical(Animal teller, Animal receiver) {  
 System.*out*.println(teller + " рассказывал в стихах " + receiver);  
}

}

public interface Pyatochok {  
 void walk();  
 void think();  
}

public class Pyato extends Animal implements Pyatochok {  
 public Pyato(String name) {  
 super(name);  
 }  
  
 @Override  
 public void walk() {  
 System.*out*.println(this + " " + TypeMoves.*WALK*.getMove());

}  
  
 @Override  
 public void say(String whatToSay) {  
 System.*out*.println(whatToSay);  
 }  
  
 @Override  
 public void think() {  
 System.*out*.println(this + " размышлял ");  
 }  
}

public enum GrowthStage {  
 *NUTSEED*("желудь"),  
 *SPROUT*("росток"),  
 *TREE*("дерево");  
  
 private final String growthStage;  
  
 GrowthStage(String growthStage) { this.growthStage = growthStage; }  
 public String getGrowthStage() { return growthStage; }  
}

public enum TypeMoves {  
 *WALK*("продолжал прогулку"),  
 *RUN*("побежал"),  
 *STOP\_AND\_STAY\_STILL*("остановился");  
  
 private final String move;  
  
 TypeMoves(String move) { this.move = move; }  
 public String getMove() {return move;}  
}

**Вывод программы:**

Винни-Пух продолжал прогулку

Пяточок продолжал прогулку

Винни-Пух рассказывал в стихах Пяточок

"неважно, чем я занят, так как я толстеть не стану, а ведь я толстеть не стану"

Пяточок размышлял скоро желудь вырастет?

**Вывод:**

Благодаря этой лабораторной работе я применила свои знания в ООП на практике, научилась использовать перечисляемы тип данных (enum), интерфесы, абстрактные классы. Изучила стандартные методы класса Object (equals(), toString() и hashCode() ) и научилась их переопределять. Познакомилась с главными принципами объектно-ориентированного программирования SOLID.