Университет ИТМО

Лабораторная работа №3
По программирование

Преподаватель: Горбунов Михаил Витальевич
Выполнила: Зайцева Ольга Олеговна
Группа: P3110
Вариант: 10189

Санкт-Петербург
2020

Задание

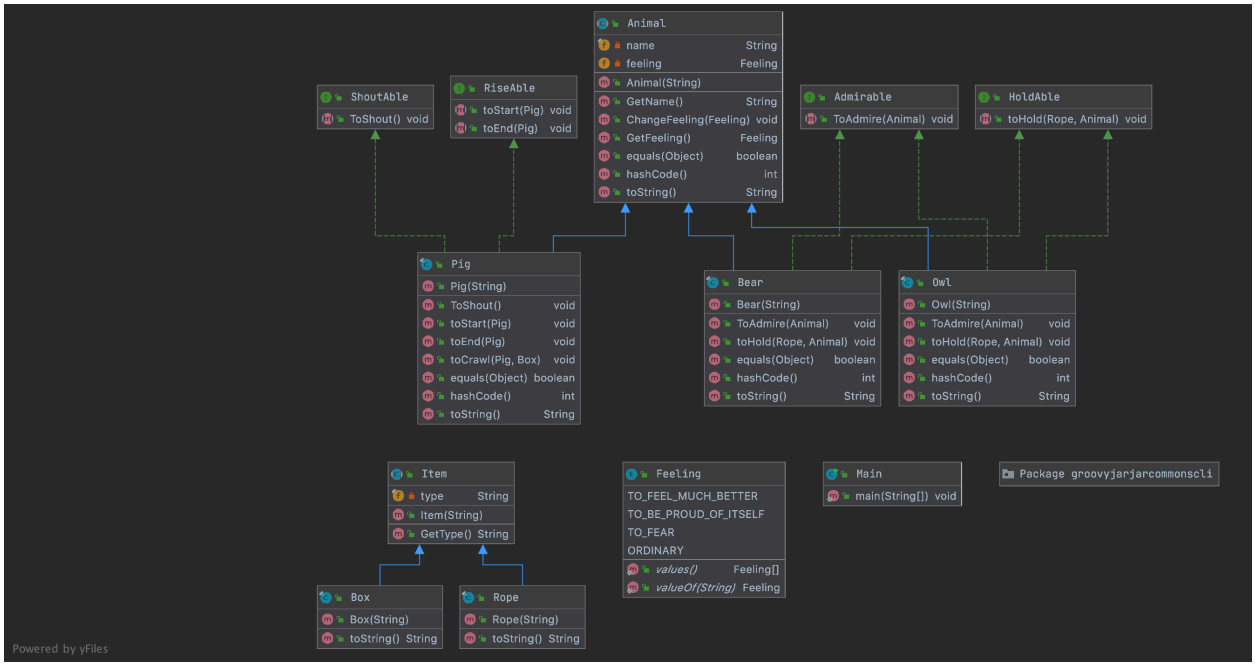
Описание предметной области, по которой должна быть построена объектная модель:

> Тут Пятачок почувствовал себя гораздо лучше, и, когда все было готово и он начал плавно подниматься к потолку, его охватила такая гордость, что он, конечно, закричал бы: "Вы поглядите на меня", если бы не опасался, что Пух и Сова, залюбовавшись им, выпустят свой конец веревки. Вскоре подъем был окончен. Пятачок открыл ящик и пролез внутрь, затем, отвязавшись, он начал протискиваться в щель, сквозь которую в добрые старые времена, когда входные двери были входными дверями, входило, бывало, много нежданных писем, которые хозяйка вдруг получала от некоей Савы.

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).

Диаграмма классов



Main.java

```java
public class Main {
    public static void main(String[] args) {
        Bear pooh = new Bear( name: "Винни Пух");
        Pig piglet = new Pig( name: "Пятачок");
        Owl owl = new Owl( name: "Сова");
        Rope rope = new Rope( type: "веревка");
        Box box = new Box( type: "ящик");

        piglet.ChangeFeeling(Feeling.TO_FEEL_MUCH_BETTER);
        pooh.toHold(rope, pooh);
        owl.toHold(rope, owl);
        piglet.toStart(piglet);
        piglet.ChangeFeeling(Feeling.TO_BE_PROUD_OF_ITSELF);
        pooh.ToAdmire(piglet);
        owl.ToAdmire(piglet);
        piglet.ChangeFeeling(Feeling.TO_FEAR);
        if (piglet.GetFeeling() == Feeling.TO_FEAR){
            System.out.println("Пятачок побоялся и не закричал");
        }
        else{
            System.out.println("Пятачок прокричал: ");
            piglet.ToShout();
        }
        piglet.toEnd(piglet);
        piglet.toCrawl(piglet, box);
    }
}
```

Admirable.java

```java
public interface Admirable {
    void ToAdmire(Animal animal);
}
```

Animal.java

```java
import java.util.Objects;
public abstract class Animal {
    private final String name;
    private Feeling feeling = Feeling.ORDINARY;

    public Animal(String name) { this.name = name; }

    public String GetName() { return name; }

    public void ChangeFeeling(Feeling feeling){
        this.feeling = feeling;
        System.out.println(GetName() + " теперь чувствует " + feeling.toString());
    }
    public Feeling GetFeeling(){
        return feeling;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Animal animal = (Animal) o;
        return Objects.equals(name, animal.name) && feeling == animal.feeling;
    }
    @Override
    public int hashCode() {
        return Objects.hash(name, feeling);
    }
    @Override
    public String toString() {
        return "Human{" +
                "name='" + name + '\'' +
                ", status=" + feeling +
                '}';
    }
}
```

Bear.java

```java
import java.util.Objects;
public final class Bear extends Animal implements Admirable, HoldAble{

    public Bear (String name){
        super(name);
        System.out.println("Создан медведь с именем " + name);
    }

    @Override
    public void ToAdmire(Animal animal){
        System.out.println(GetName() + " залюбовался " + animal.GetName());
    }
    @Override
    public void toHold(Rope rope, Animal animal){
        System.out.println(animal.GetName() + " держит " + rope.GetType());
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Bear bear = (Bear) o;
        return Objects.equals(GetName(), bear.GetName());
    }

    @Override
    public int hashCode() {
        return Objects.hash(GetName());
    }

    @Override
    public String toString() {
        return "Bear";
    }
}
```

Box.java

```java
public final class Box extends Item{
    public Box (String type){
        super(type);
        System.out.println("Создана " + type);
    }
    @Override
    public String toString() {
        return "Box";
    }
}
```

Feeling.java

```java
public enum Feeling {
    TO_FEEL_MUCH_BETTER,
    TO_BE_PROUD_OF_ITSELF,
    TO_FEAR,
    ORDINARY
}
```

HoldAble.java

```java
public interface HoldAble {
    void toHold(Rope rope, Animal animal);
}
```

Item.java

```java
public abstract class Item {
    private final String type;

    public Item(String type){
        this.type = type;
    }

    public String GetType(){
        return type;
    }
}
```

Owl.java

```java
import java.util.Objects;
public final class Owl extends Animal implements Admirable, HoldAble{

    public Owl (String name){
        super(name);
        System.out.println("Создана сова с именем " + name);
    }

    @Override
    public void ToAdmire(Animal animal){

        System.out.println(GetName() + " залюбовался " + animal.GetName());
    }
    @Override
    public void toHold(Rope rope, Animal animal) { System.out.println(animal.GetName() + " держит " + rope.GetType()); }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Owl owl = (Owl) o;
        return Objects.equals(GetName(), owl.GetName());
    }

    @Override
    public int hashCode() { return Objects.hash(GetName()); }

    @Override
    public String toString() { return "Owl"; }
}
```

Pig.java

```java
import java.util.Objects;

public final class Pig extends Animal implements ShoutAble, RiseAble{
    public Pig (String name){
        super(name);
        System.out.println("Создан поросенок с именем " + name);
    }
    @Override
    public void ToShout() { System.out.println("Вы поглядите на меня"); }
    @Override
    public void toStart(Pig pig) { System.out.println(pig.GetName() + " начал подъем"); }
    @Override
    public void toEnd(Pig pig) { System.out.println(pig.GetName() + " закончил подъем"); }
    public void toCrawl(Pig pig, Box box) { System.out.println(pig.GetName() + " полез в " + box.GetType()); }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Pig pig = (Pig) o;
        return Objects.equals(GetName(), pig.GetName());
    }

    @Override
    public int hashCode() {
        return Objects.hash(GetName());
    }

    @Override
    public String toString() { return "Pig"; }
}
```

Riseable.java

```java
public interface RiseAble {
    void toStart(Pig pig);
    void toEnd(Pig pig);
}
```

Rope.java

```java
public final class Rope extends Item{

    public Rope (String type){
        super(type);
        System.out.println("Создана " + type);
    }
    @Override
    public String toString() {
        return "Rope";
    }

}
```

ShoutAble.java

```java
public interface ShoutAble {
    void ToShout();
}
```

Выводы:

В ходе выполнения данной лабораторной работы я ознакомилась с созданием объектно-ориентированных моделей в Java.