Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет ПИиКТ



ОТЧЁТ

По лабораторной работе №3

Вариант 3985

Работу выполнил:

Андрейченко Леонид Вадимович

Группа P3130

Преподаватель:

Николаев Владимир Вячеславович

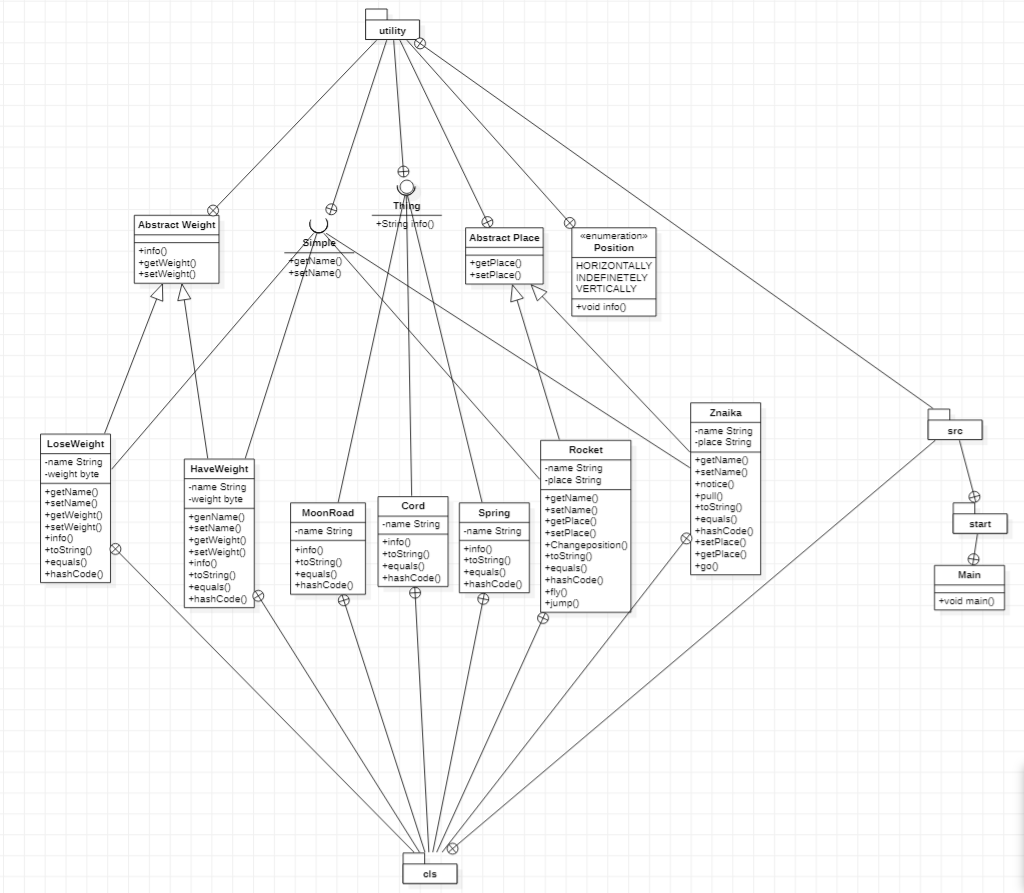
Санкт – Петербург

2021

Описание предметной области

Как известно, все предметы, теряя вес, обычно поднимаются вверх (если они, конечно, не закреплены). Ведь, находясь под действием силы тяжести, каждый предмет как бы сжимается или сплющивается хотя бы на самую ничтожную величину. Но как только предмет потеряет вес, он разжимается, выпрямляется, в результате чего отталкивается, как пружина, от поверхности, на которой до этого неподвижно стоял. Заметив, что ракета поднялась на достаточную высоту, Знайка потихонечку потянул за шнур и не спеша зашагал по лунной дорожке. Ракета приняла горизонтальное положение и послушно поплыла над поверхностью Луны. Правда, по временам она опускалась, но, едва коснувшись поверхности Луны, отталкивалась от нее и снова поднималась вверх.

UML диаграмма



Исходный код

Start.Main

package start;

import cls.\*;

import utility.Position;

public class Main {

public static void main(String[] args) {

Znaika znaika = new Znaika("Знайка","Луны");

Rocket rocket = new Rocket("Ракета","Луны");

HaveWeight haveWeight = new HaveWeight("Предметы, имеющие вес", (byte) 1);

LoseWeight loseWeight= new LoseWeight("Предметы, теряющие вес", (byte) -1);

Spring spring = new Spring();

Cord cord = new Cord();

MoonRoad moonRoad = new MoonRoad();

Position position = Position.HORIZONTALLY;

loseWeight.Info(spring.info());

haveWeight.Info("");

znaika.notice(" что ракета поднялась на достаточную высоту,");

znaika.pull(cord.info());

znaika.go(moonRoad.info());

rocket.ChangePosition(position.getInfo());

rocket.fly();

rocket.jump();

}

}

Utility.AbstractPlace

package utility;  
  
public abstract class AbstractPlace {  
  
 public void getPlace (){}  
 public void setPlace(String place){}  
}

Utility.AbstractWeight

package utility;  
  
public abstract class AbstractWeight {  
  
 public void Info(String s){}  
 public void getWeight (){}  
 public void setWeight(byte place){}  
  
}

Utility.Simple

package utility;  
  
public interface Simple {  
 void getName ();  
 void setName(String name);  
}

Utility.Thing

package utility;  
  
public interface Thing {  
 public String info();  
}

Utility.Position

package utility;  
  
public enum Position {  
 INDEFINETELY("неопределенное"), VERTICALLY("вертикальное"), HORIZONTALLY("горизонтальное");  
  
 private String info;  
  
 Position(String s){  
 info = s;  
 }  
  
 public String getInfo() {  
 return info;  
 }  
}

Cls.Cord

package cls;  
  
import utility.Thing;  
  
import java.util.Objects;  
  
public class Cord implements Thing {  
 private String name = "шнур";  
  
 @Override  
 public String info() {  
 return name;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Cord cord = (Cord) o;  
 return Objects.equals(name, cord.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name);  
 }  
  
 @Override  
 public String toString() {  
 return "Cord{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
}

Cls.HaveWeight

package cls;  
  
import utility.AbstractWeight;  
import utility.Simple;  
  
import java.util.Objects;  
  
public class HaveWeight extends AbstractWeight implements Simple {  
  
 private String name;  
 private byte weight;  
  
 public HaveWeight(String name, byte weight){  
 this.name=name;  
 this.weight=weight;  
 }  
  
 @Override  
 public void getName() {  
 System.out.print(name);  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name=name;  
 }  
  
 @Override  
 public void getWeight() {  
 System.out.println(weight);  
 }  
  
 @Override  
 public void setWeight(byte weight) {  
 this.weight=weight;  
 }  
  
 @Override  
 public void Info(String s){  
 getName();  
 System.out.print(", находясь под действием силы тяжести, как бы сжимаются или сплющиваются хотя бы на самую ничтожную величину.");  
 System.out.println("");  
 }  
  
 @Override  
 public String toString() {  
 return "HaveWeight{" +  
 "name='" + name + '\'' +  
 ", weight=" + weight +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 HaveWeight that = (HaveWeight) o;  
 return weight == that.weight &&  
 Objects.equals(name, that.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name, weight);  
 }  
}

Cls.LoseWeight

package cls;  
  
import utility.AbstractWeight;  
import utility.Simple;  
  
import java.util.Objects;  
  
public class LoseWeight extends AbstractWeight implements Simple {  
  
 private String name;  
 private byte weight;  
  
 public LoseWeight(String name, byte weight){  
 this.name=name;  
 this.weight=weight;  
 }  
  
 @Override  
 public void getName() {  
 System.out.print(name);  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name=name;  
 }  
  
 @Override  
 public void getWeight() {  
 System.out.println(weight);  
 }  
  
 @Override  
 public void setWeight(byte weight) {  
 this.weight=weight;  
 }  
  
 @Override  
 public void Info(String s){  
 getName();  
 System.out.println(", обычно поднимаются вверх (если они, конечно, не закреплены). Они разжимаются, выпрямляются, в результате чего отталкиваются, как "+s+", от поверхности, на которой до этого неподвижно стояли");  
 }  
  
 @Override  
 public String toString() {  
 return "LoseWeight{" +  
 "name='" + name + '\'' +  
 ", weight=" + weight +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 LoseWeight that = (LoseWeight) o;  
 return weight == that.weight &&  
 Objects.equals(name, that.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name, weight);  
 }  
}

Cls.MoonRoad

package cls;  
  
import utility.Thing;  
  
import java.util.Objects;  
  
public class MoonRoad implements Thing {  
 private String name = "лунной дорожке";  
  
 @Override  
 public String info() {  
 return name;  
 }  
  
 @Override  
 public String toString() {  
 return "MoonRoad{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 MoonRoad moonRoad = (MoonRoad) o;  
 return Objects.equals(name, moonRoad.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name);  
 }  
}

Cls.Spring

package cls;  
import utility.Thing;  
  
import java.util.Objects;  
  
public class Spring implements Thing {  
 private String name = "пружина";  
  
  
 @Override  
 public String info() {  
 return name;  
 }  
  
 @Override  
 public String toString() {  
 return "Spring{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Spring spring = (Spring) o;  
 return Objects.equals(name, spring.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name);  
 }  
}

Cls.Rocket

package cls;  
import utility.Simple;  
import utility.AbstractPlace;  
import java.util.Objects;  
public class Rocket extends AbstractPlace implements Simple{  
 private String name;  
 private String place;  
 public Rocket(String name, String place){  
 this.name=name;  
 this.place=place;  
 }  
  
 @Override  
 public void getName() {  
 System.out.print(name);  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name=name;  
 }  
  
 @Override  
 public void getPlace() {  
 System.out.print(place);  
 }  
  
 @Override  
 public void setPlace(String place) {  
 this.place=place;  
 }  
  
 public void ChangePosition(String s){  
 getName();  
 System.out.print(" приняла "+s+" положение");  
 }  
  
 public void fly(){  
 System.out.print(", поплыла над поверхностью ");  
 getPlace();  
 }  
  
 public void jump(){  
 System.out.println("");  
 getName();  
 System.out.print(" опускалась, но, едва коснувшись поверхности ");  
 getPlace();  
 System.out.print(", отталкивалась от нее и снова поднималась вверх.");  
 }  
  
 @Override  
 public String toString() {  
 return "Rocket{" +  
 "name='" + name + '\'' +  
 ", place='" + place + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Rocket rocket = (Rocket) o;  
 return Objects.equals(name, rocket.name) &&  
 Objects.equals(place, rocket.place);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name, place);  
 }  
}

Cls.Znaika

package cls;  
  
import utility.AbstractPlace;  
import utility.Simple;  
  
import java.util.Objects;  
  
public class Znaika extends AbstractPlace implements Simple {  
 private String name;  
 private String place;  
  
 public Znaika(String name, String place){  
 this.name=name;  
 this.place=place;  
 }  
 @Override  
 public void getName() {  
 System.out.print(name);  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name=name;  
 }  
  
 @Override  
 public void getPlace() {  
 System.out.println(place);  
 }  
  
 @Override  
 public void setPlace(String place) {  
 this.place=place;  
 }  
  
 public void notice(String s){  
 getName();  
 System.out.print(", заметил "+s);  
 }  
  
 public void pull(String s){  
 System.out.print(" потянул за "+s);  
 }  
  
 public void go(String s){  
 System.out.print(", и зашагал по "+s);  
 System.out.println("");  
 }  
   
 @Override  
 public String toString() {  
 return "Znaika{" +  
 "name='" + name + '\'' +  
 ", place='" + place + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Znaika znaika = (Znaika) o;  
 return Objects.equals(name, znaika.name) &&  
 Objects.equals(place, znaika.place);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(name, place);  
 }  
}

Результат работы

Предметы, теряющие вес, обычно поднимаются вверх (если они, конечно, не закреплены). Они разжимаются, выпрямляются, в результате чего отталкиваются, как пружина, от поверхности, на которой до этого неподвижно стояли

Предметы, имеющие вес, находясь под действием силы тяжести, как бы сжимаются или сплющиваются хотя бы на самую ничтожную величину.

Знайка, заметил что ракета поднялась на достаточную высоту, потянул за шнур, и зашагал по лунной дорожке

Ракета приняла горизонтальное положение, поплыла над поверхностью Луны

Ракета опускалась, но, едва коснувшись поверхности Луны, отталкивалась от нее и снова поднималась вверх

Выводы

Во время выполнения данной лабораторной работы я научился применять принципы SOLID на практике, подробнее разобрал интерфейсы, абстрактные классы и перечисления. Не выспался.