Университет ИТМО

Факультет систем управления и робототехники

Программирование

Лабораторная работа №3 Вариант №314333

Преподаватель: Сорокин Роман Борисович

Выполнил: Клещев Глеб Сергеевич

Группа:

R3142

Текст задания

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

Они снова отправились искать работу, но в этот день их поиски не увенчались успехом. Козлик сказал, что это ничего, так как скоро откроется "Веселый балаганчик" и можно будет попытаться подзаработать там. Явившись на улицу, где было уже знакомое им увеселительное заведение, Незнайка и Козлик увидели, что вчерашний коротышка выздоровел и уже стоял на помосте, увертываясь от летевших в лицо мячей. Правда, под глазом у него красовался большой синяк, но коротышка, по-видимому, привык не обращать внимания на подобные пустяки. Расчеты Козлика оказались верными. Скоро действительно кто-то запустил мяч с такой силой, что коротышка не успел увернуться. Удар на этот раз пришелся по другому глазу. Схватившись рукой за подбитый глаз и заливаясь слезами от боли, бедняга, невзвидя света, побежал поскорей домой. Козлику было жалко несчастного коротышку, но вместе с тем он был рад, что теперь сможет заработать немного денег.

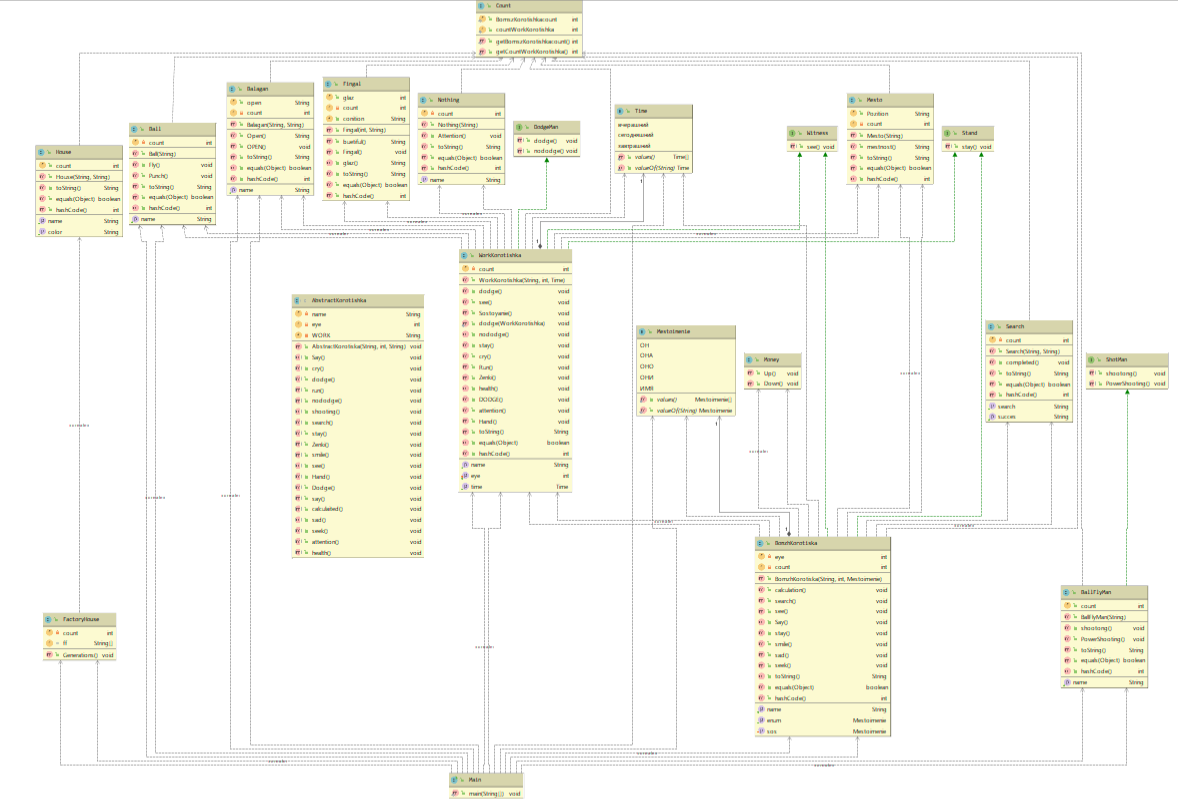
**Программа должна удовлетворять следующим требованиям:**

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.

Диаграмма классов



Исходный код программы

Main.java:

public class Main {  
 public static void main(String[] args) {  
 BomzhKorotiska Neznayka= new BomzhKorotiska("Незнайка", 2, Mestoimenie.*ИМЯ*);  
 BomzhKorotiska Kozlik= new BomzhKorotiska("Козлик", 2, Mestoimenie.*ИМЯ*);  
 Balagan balagan= new Balagan ("Закрыт", "Веселый балаганчик");  
 WorkKorotishka Korotishka= new WorkKorotishka("Коротышка", 1, Time.*вчерашний*);  
 BallFlyMan NN= new BallFlyMan("Кто-то");  
 Ball ball= new Ball("мяч");  
 FactoryHouse zavod =new FactoryHouse();  
 Neznayka.setSos(Mestoimenie.*ОНИ*);  
 Neznayka.search();  
 Neznayka.seek();  
 *//Kozlik.setSos(Mestoimenie.ОН);* Kozlik.Say();  
 balagan.OPEN();  
 Neznayka.stay();  
 Neznayka.see();  
 zavod.Generations();  
 Neznayka.see();  
 Korotishka.Sostoyanie();  
 Korotishka.health();  
 Korotishka.stay();  
 Korotishka.DODGE();  
 Korotishka.Zenki();  
 Korotishka.attention();  
 while (Korotishka.getEye() >0) {  
 Double a= Math.*random*();  
 if (a> 0.5) {  
 NN.shootong();  
 ball.Fly();  
 WorkKorotishka.*dodge*(Korotishka);  
 } else {  
 Kozlik.calculation();  
 NN.shootong();  
 NN.PowerShooting();  
 Korotishka.nododge();  
 ball.Punch();  
 Korotishka.setEye(Korotishka.getEye()-1);  
 Korotishka.Zenki();  
 Korotishka.Hand();  
 Korotishka.cry();  
 }  
 }  
 Korotishka.see();  
 Korotishka.Run();  
 Kozlik.sad();  
 Kozlik.smile();}  
}

AbstractKorotishka.jara

abstract class AbstractKorotishka {  
 private String name;  
 private int eye;  
 private String WORK;  
 public void AbstractKorotiska(String name, int eye, String WORK) {  
   
 }  
 public abstract void Say();  
 public abstract void cry();  
 public abstract void dodge();  
 public abstract void run();  
 public abstract void nododge();  
 public abstract void shooting();  
 public abstract void search();  
 public abstract void stay();  
 public abstract void Zenki();  
 public abstract void smile();  
 public abstract void see();  
 public abstract void Hand();  
 public abstract void Dodge();  
 public abstract void say();  
 public abstract void calculaited();  
 public abstract void sad();  
 public abstract void seek();  
 public abstract void attention();  
 public abstract void health();  
  
}

Balagan.java

import java.util.Objects;  
  
public class Balagan {  
 public String open, name;  
 private int count;  
 public Balagan(String open, String name) {  
 this.open=open;  
 this.name=name;  
 count = Count.*getCountWorkKorotishka*();  
 }  
 public String Open() {  
 return open;  
 }  
 public String getName() {  
 return name;  
 }  
 public void OPEN() {  
 if (Open()=="Закрыт") {  
 System.*out*.print(getName() + " скоро откроется. Можно будет попытаться подзаработать там. ");  
 System.*out*.println("");  
 } else {  
 System.*out*.println(getName() + Open());  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Балаганчик{" +  
 "открыт='" + open + '\'' +  
 ", имя='" + name + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Balagan balagan = (Balagan) o;  
 return Objects.*equals*(open, balagan.open) &&  
 Objects.*equals*(name, balagan.name);  
 }  
  
 @Override  
 public int hashCode() {  
  
 return (count<<2)\*(count<<2)-1;  
 }  
}

Ball.java

import java.util.Objects;  
  
public class Ball {  
 private String name;  
 private int count;  
 public Ball(String name){  
 this.name=name;  
 count = Count.*getBomszKorotishkacount*();  
 }  
 public void Fly() {  
 System.*out*.println(" мяч летит");  
 }  
 public void Punch() {  
 System.*out*.println("Удар пришелся по другому глазу");  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 @Override  
 public String toString() {  
 return "мяч{" +  
 "имя='" + name + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Ball ball = (Ball) o;  
 return Objects.*equals*(name, ball.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2);  
 }  
}

### BallFlyMan.java

import java.util.Objects;  
  
public class BallFlyMan implements ShotMan {  
 private String name;  
 public int count;  
 public BallFlyMan(String name) {  
 this.name=name; count = Count.*getBomszKorotishkacount*();  
 }  
 public String getName() {  
 return name;  
 }  
 public void shootong() {  
 System.*out*.print(getName()+" кидает мяч");  
 }  
 public void PowerShooting() {System.*out*.println(" с такой силой ");}  
  
 @Override  
 public String toString() {  
 return "Кидающий мяч{" +  
 "имя='" + name + '\'' +  
 ", счетчик=" + count +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 BallFlyMan that = (BallFlyMan) o;  
 return count == that.count &&  
 Objects.*equals*(name, that.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2)+(count<<2)\*(count<<2);  
 }  
}

### BomzhKorotiska.java

import java.util.Objects;  
  
public class BomzhKorotiska implements Witness, Stand {  
 private String name;  
 private int eye;  
 private Mestoimenie sos;  
 private int count;  
  
 public BomzhKorotiska(String name, int eye, Mestoimenie sos) {  
 this.sos=sos;  
 this.eye=eye;  
 this.name=name;  
 count = Count.*getBomszKorotishkacount*();  
}  
 public String getName() {  
 return this.name;  
 }  
  
 public void setSos(Mestoimenie sos) {  
 this.sos = sos;  
 }  
 public void calculation() {  
 System.*out*.println("Рассчеты "+getName()+ "a оказались верными");  
 }  
  
 public Mestoimenie getEnum() {  
 return sos;  
}  
 public void search() {  
 switch (sos) {  
 case *ОНИ*:  
 System.*out*.print(getEnum() + " снова отправились искать работу, но в этот день их ");  
 break;  
 case *ОНА*:  
 System.*out*.println(getEnum() + " снова отправились искать работу, но в этот день её");  
 break;  
 case *ОН*:  
 System.*out*.println(getEnum() + " снова отправились искать работу, но в этот день его");  
 break;  
 case *ОНО*:  
 System.*out*.println(getEnum() + " снова отправились искать работу, но в этот день его");  
 break;  
 case *ИМЯ*:  
 System.*out*.println(getName() + " снова отправились искать работу, но в этот день "+ getName());  
 break;  
 }  
 }  
 public void see () {  
 if (this.eye > 0) {  
 switch (sos) {  
 case *ОНИ*:  
 System.*out*.print(getEnum() + " видят ");  
 break;  
 case *ОНА*:  
 System.*out*.print(getEnum() + " видит ");  
 break;  
 case *ОН*:  
 System.*out*.print(getEnum() + " видит ");  
 break;  
 case *ОНО*:  
 System.*out*.print(getEnum() + " видит ");  
 break;  
 case *ИМЯ*:  
 System.*out*.print(getName() + " видит ");  
 break;  
 }  
 }  
 if (this.eye == 0) {  
 System.*out*.println("Ничего не вижу");  
 }  
 }  
 public void Say() {  
 switch (sos) {  
 case *ОНИ*:  
 System.*out*.print(getEnum() + "говорят: это ничего,");  
 break;  
 case *ОНА*:  
 System.*out*.print(getEnum() + " говорит: это ничего, ");  
 break;  
 case *ОН*:  
 System.*out*.print(getEnum() + " говорит: это ничего, ");  
 break;  
 case *ОНО*:  
 System.*out*.print(getEnum() + " говорит: это ничего, ");  
 break;  
 case *ИМЯ*:  
 System.*out*.print(getName() + " говорит: это ничего, ");  
 break;  
 }  
 }  
 public void stay() {  
 Mesto mesto = new Mesto(" улицу,где было знакомое увеселительное заведение");  
 switch (sos) {  
 case *ОНИ*:  
 System.*out*.println(getEnum() + " явились" + mesto.mestnost());  
 break;  
 case *ОНА*:  
 System.*out*.println(getEnum() + " явились" + mesto.mestnost());  
 break;  
 case *ОН*:  
 System.*out*.println(getEnum() + " явились" + mesto.mestnost());  
 break;  
 case *ОНО*:  
 System.*out*.println(getEnum() + " явились" + mesto.mestnost());  
 break;  
 case *ИМЯ*:  
 System.*out*.println(getName() + " явились" + mesto.mestnost());  
 break;  
 }  
 }  
 public void smile() {  
 Money dengi = new Money();  
 switch (sos) {  
 case *ОНИ*:  
 System.*out*.print(getEnum() + " обрадовались, ");  
 break;  
 case *ОНА*:  
 System.*out*.print(getEnum() + " обрадовалась, ");  
 break;  
 case *ОН*:  
 System.*out*.print(getEnum() + " обрадовался, ");  
 break;  
 case *ОНО*:  
 System.*out*.print(getEnum() + " обрадовался, ");  
 break;  
 case *ИМЯ*:  
 System.*out*.print(getName() + " обрадовался, ");  
 break;  
 }  
  
 dengi.Up();  
 }  
 public void sad() {  
 WorkKorotishka Korotiska;  
 Korotiska = new WorkKorotishka("коротышку", 0, Time.*вчерашний* );  
 switch (sos) {  
 case *ОНИ*:  
 System.*out*.print(getEnum() + " пожалели, "+ Korotiska.getName());  
 break;  
 case *ОНА*:  
 System.*out*.print(getEnum() + " пожалела, "+ Korotiska.getName());  
 break;  
 case *ОН*:  
 System.*out*.print(getEnum() + " пожалел, "+ Korotiska.getName());  
 break;  
 case *ОНО*:  
 System.*out*.print(getEnum() + " пожалело, "+ Korotiska.getName());  
 break;  
 case *ИМЯ*:  
 System.*out*.println(getName() + "у было жалко, "+ Korotiska.getName());  
 break;  
 }  
 }  
 public void seek() {  
 Search search= new Search("поиски", "не увенчались успехом");  
 search.completed();  
 }  
  
 @Override  
 public String toString() {  
 return "БомжКоротышка{" +  
 "имя='" + name + '\'' +  
 ", глаз=" + eye +  
 ", местоимение=" + sos +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 BomzhKorotiska that = (BomzhKorotiska) o;  
 return eye == that.eye &&  
 count == that.count &&  
 Objects.*equals*(name, that.name) &&  
 sos == that.sos;  
 }  
  
 @Override  
 public int hashCode() {  
 return (eye<<1)+(count<<2);  
 }  
}

Count.java

public class Count {  
 public static int *BomszKorotishkacount* = 0;  
  
 public static int getBomszKorotishkacount() {  
 *BomszKorotishkacount*++;  
 return *BomszKorotishkacount*-1;  
 }  
 public static int *countWorkKorotishka*=0;  
  
 public static int getCountWorkKorotishka() {  
 *countWorkKorotishka*++;  
 return *countWorkKorotishka*-1;  
 }  
  
}

DodgeMan.java

public interface DodgeMan {  
 public void dodge();  
 public void nododge();  
}

import java.util.Arrays;  
  
public class FactoryHouse {  
 private int count;  
 String[] ff = new String[] {"черного","белого","красного","синего"};  
 public void Generations() {  
 System.*out*.print("Улица состоит из :") ;  
 for (int i = 0; i < 4; i++){  
 System.*out*.print(new House("zdanie" + i,ff[i]).getColor() + " ");  
 }  
 System.*out*.println(" домов");  
 }  
}

FactoryHouse.java

import java.util.Arrays;  
  
public class FactoryHouse {  
 private int count;  
 String[] ff = new String[] {"черного","белого","красного","синего"};  
 public void Generations() {  
 System.*out*.print("Улица состоит из :") ;  
 for (int i = 0; i < 4; i++){  
 System.*out*.print(new House("zdanie" + i,ff[i]).getColor() + " ");  
 }  
 System.*out*.println(" домов");  
 }  
}

Fingal.java

import java.util.Objects;  
  
public class Fingal {  
 public int glaz;  
 private int count;  
 public String conition;  
 public Fingal(int glaz, String condition) {  
 this.glaz=glaz;  
 this.conition=condition;  
 count = Count.*getCountWorkKorotishka*();  
 }  
 public String buetiful() {  
 return " у него красовался ";  
 }  
 public void Fingal() {  
 if (glaz==1) {  
 System.*out*.println(" Под глазом"+buetiful()+ "Большой синяк");  
 }  
 if (glaz==0) {  
 System.*out*.println("Большие синяки под глазами");  
 }  
 if (glaz==2) {  
 System.*out*.println("нет синяков");  
 }  
  
 }  
 public String glaz() {  
 return conition + " глаз";  
 }  
 @Override  
 public String toString() {  
 return "FФингал{" +  
 "глаз=" + glaz +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Fingal fingal = (Fingal) o;  
 return glaz == fingal.glaz;  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2)+(glaz<<1);  
 }  
}

House.java

import java.util.Objects;  
  
public class House {  
 public String name;  
 public String color;  
 public int count;  
 public House(String name, String color) {  
 this.name=name;  
 this.color=color;  
 count = Count.*getCountWorkKorotishka*();  
 }  
 public String getName() {  
 return name;  
 }  
 public String getColor() {  
   
 return color;  
 }  
  
 @Override  
 public String toString() {  
 return "Дом{" +  
 "имя='" + name + '\'' +  
 ", цвет='" + color + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 House house = (House) o;  
 return Objects.*equals*(name, house.name) &&  
 Objects.*equals*(color, house.color);  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2)\*2;  
 }  
}

Mesto.java

import java.util.Objects;  
  
public class Mesto {  
 public String Pozition;  
 private int count;  
 public Mesto(String Pozition) {  
 this.Pozition=Pozition; count = Count.*BomszKorotishkacount*;  
 }  
 public String mestnost() {  
 return Pozition;  
 }  
  
 @Override  
 public String toString() {  
 return "Место{" +  
 "Позиция='" + Pozition + '\'' +  
 ", счетчик=" + count +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Mesto mesto = (Mesto) o;  
 return count == mesto.count &&  
 Objects.*equals*(Pozition, mesto.Pozition);  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2)-5;  
 }  
}

Mestoimenie.java

public enum Mestoimenie {  
 *ОН*,  
 *ОНА*,  
 *ОНО*,  
 *ОНИ*,  
 *ИМЯ*}

Money.java

public class Money {  
 public void Up() {  
 System.*out*.print(" заработает немного Денег");  
 }  
 public void Down() {  
 System.*out*.print(" потратит немного деньги");  
 }  
  
}

Nothing.java

import java.util.Objects;  
  
public class Nothing {  
 private String name;  
 private int count;  
 public Nothing(String name){  
 this.name=name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void Attention(){  
 System.*out*.println("Не обращать внимание на "+getName());  
 count = Count.*getCountWorkKorotishka*();  
 }  
  
 @Override  
 public String  
 toString() {  
 return "Пустяк{" +  
 "имя='" + name + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Nothing nothing = (Nothing) o;  
 return Objects.*equals*(name, nothing.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2)+3+ (count<<2)\*(count<<2);  
 }  
}

Search.java

import java.util.Objects;  
  
public class Search {  
 private String search;  
 private String succes;  
 private int count;  
 public Search(String search, String succes) {  
 this.search=search;  
 this.succes=succes;  
 count = Count.*getCountWorkKorotishka*();  
 }  
 public void completed() {  
 System.*out*.println(getSearch()+" " + getSucces());  
 }  
 public String getSucces() {  
 return succes;  
 }  
  
 public String getSearch() {  
 return search;  
 }  
  
 @Override  
 public String  
 toString() {  
 return "Поиск{" +  
 "искать='" + search + '\'' +  
 ", успешность='" + succes + '\'' +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Search search1 = (Search) o;  
 return Objects.*equals*(search, search1.search) &&  
 Objects.*equals*(succes, search1.succes);  
 }  
  
 @Override  
 public int hashCode() {  
 return (count<<2)\*(count<<2)-4;  
 }  
}

ShotMan.java

public interface ShotMan {  
 public void shootong();  
 public void PowerShooting();  
}

Stand.java

public interface Stand {  
 public void stay();  
}

Time.java

public enum Time {  
 *вчерашний*,  
 *сегодняшний*,  
 *завтрашний*}

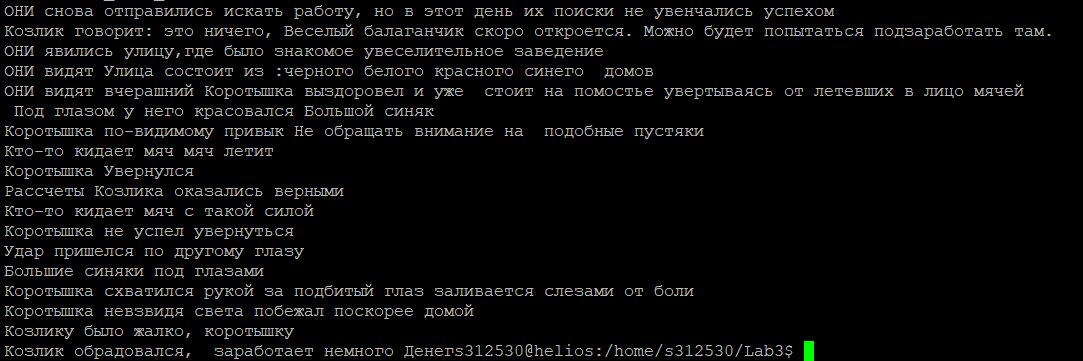
Witness.java

public interface Witness {  
 public void see();  
}

WorkKorotiska.java

import java.util.Objects;  
  
public class WorkKorotishka implements DodgeMan, Witness, Stand {  
 private String name;  
 private int eye;  
 private int count;  
 private Time time;  
  
 public WorkKorotishka(String name, int eye, Time time) {  
 this.eye = eye;  
 this.name = name;  
 this.time = time;  
 count = Count.*getCountWorkKorotishka*();  
 }  
 public String getName() {  
 return this.name;  
 }  
 public Time getTime() {return time;}  
 public int getEye() {  
 return eye;  
 }  
  
 public void setEye(int eye) {  
 this.eye = eye;  
 }  
 public void dodge() {  
System.*out*.println(getName()+ "уворачивается");  
 }  
  
 public void see () {  
 if (this.eye > 0) {  
 System.*out*.println(getName()+" видит мяч");  
 }  
 if (this.eye == 0) {  
 System.*out*.print(getName()+" невзвидя света");  
 }  
 }  
 public void Sostoyanie() {  
 System.*out*.print(getTime()+" "+ getName());  
 }  
 public static void dodge(WorkKorotishka workKorotishka) {  
 System.*out*.println(workKorotishka.getName()+" Увернулся");  
 }  
 public void nododge() {  
 System.*out*.println(getName()+" не успел увернуться");  
 }  
 public void stay() {  
 Mesto mesto = new Mesto("помостье");  
 mesto.mestnost();  
 System.*out*.print(" стоит на " + mesto.mestnost());  
 }  
 public void cry() {  
 System.*out*.println(" заливается слезами от боли");  
 }  
 public void Run() {  
 Balagan home =new Balagan(" открыто", "домой");  
 System.*out*.println(" побежал поскорее "+ home.getName());  
 }  
 public void Zenki() {  
 Fingal oko =new Fingal(this.eye, "подбитый");  
 oko.Fingal();  
 }  
 public void health() {  
 System.*out*.print(" выздоровел и уже ");  
 }  
 public void DODGE() {  
 Ball ball =new Ball("мячей");  
 System.*out*.println(" увертываясь от летевших в лицо "+ball.getName());  
 }  
 public void attention(){  
 Nothing nothing=new Nothing(" подобные пустяки");  
 System.*out*.print(getName()+" по-видимому привык ");  
 nothing.Attention();  
 }  
 public void Hand() {  
 Fingal oko =new Fingal(this.eye, "подбитый");  
 System.*out*.print(getName()+ " схватился рукой за " +oko.glaz());  
 }  
 @Override  
 public String toString() {  
 return "Рабочий коротышка{" +  
 "имя='" + name + '\'' +  
 ", глаз=" + eye +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 WorkKorotishka that = (WorkKorotishka) o;  
 return eye == that.eye &&  
 Objects.*equals*(name, that.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return (eye<<1)+(count<<2);  
 }  
}

# Пример работы программы



Выводы

В процессе выполнения лабораторной работы я ознакомился с принципами SOLID, ознакомился с абстрактными классами, интерфейсами и перечислениями