

Факультет программной инженерии и

компьютерной техники

Лабораторная работа №4

**Основы объектно-ориентированного программирования**

Вариант №34082

Выполнила Громилова Мария Дмитриевна,

группа P3130

Преподаватель Блохина Елена Николаевна

г. Санкт-Петербург 2021 г.

**Оглавление:**

[Задание к лабораторной работе: 2](#_Toc90930106)

[Диаграмма классов реализованной объектной модели: 3](#_Toc90930107)

[Исходный код: 3](#_Toc90930108)

[Результаты работы программы: 10](#_Toc90930109)

[Вывод: 11](#_Toc90930110)

# Задание к лабораторной работе:

Доработать программу из лабораторной работы №3, обновив реализацию объектной модели в соответствии с новой версией описания предметной области.

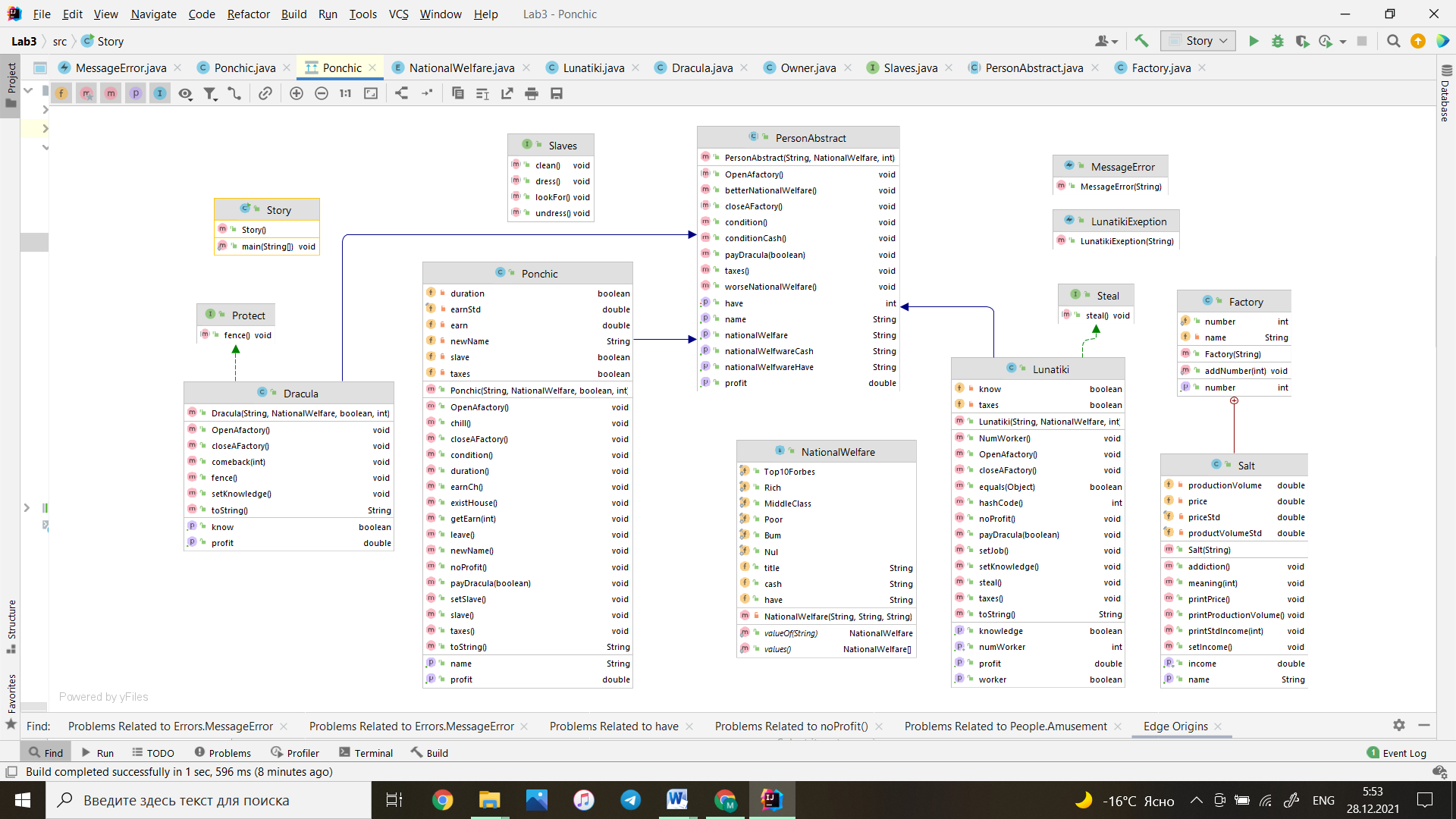
Текст задания:

Жил теперь Пончик в полное свое удовольствие, как и все остальные  
лунные богачи, и даже назывался он теперь не просто Пончик, а господин  
Понч. Из гостиницы он переехал в собственный дом, завел себе слуг, которые  
одевали его и раздевали, убирали у него в комнатах, смотрели за домом. От  
нечего делать он по целым дням просиживал в ресторанах, ел там самые  
вкусные кушанья, а в промежутках между едой околачивался на берегу залива  
и вертелся на чертовых водяных колесах или на морском параболоиде.

Такое беспечное существование Пончика длилось все же недолго. Многим лунатикам удалось вскоре проведать, где он берет кристаллы соли для своего завода. Это привело к тому, что неподалеку от берега один за другим начали возникать небольшие соляные заводики. Каждый, кому удалось сберечь достаточную сумму денег, затрачивал ее на устройство такого заводика и начинал приумножать свое богатство. В результате соли с каждым днем добывалось все больше, а продавалась она все дешевле и уже не приносила таких барышей, как вначале. Если прежде Пончик, затратив на оплату рабочего фертинг, получал взамен двадцать фертингов прибыли, то теперь никто не мог выколотить из рабочего больше одного фертинга. Однако и это было очень выгодно, в силу чего соляной промысел продолжал развиваться. Дела пошли значительно хуже, когда в Лос-Паганос вернулся из поездки крупнейший землевладелец Дракула, которому принадлежало все морское побережье, начиная от Лос-Паганоса вплоть до самого ЛосСвиноса. Узнав, что какие-то неизвестные личности растаскивают лежавшие на побережье кристаллы для переработки их на соляных заводах, он велел обнести побережье забором, а кристаллы давать только тем, кто будет за них платить. Таким образом, половина прибыли, которую получали владельцы соляных заводов, теперь стала попадать в карманы землевладельца Дракулы.

Соль стала продаваться по такой низкой цене, что Пончику и остальным мелким хозяйчикам приходилось расходовать на  
приобретение соляных кристаллов и оплату рабочих гораздо больше денег, чем  
они выручали от продажи своей продукции. Мелкие соляные заводики стали  
закрываться один за другим. Пончик держался дольше других. Чтобы  
как-нибудь сводить концы с концами, он продал свой дом, продал новое  
помещение для завода, которое едва успел построить, но все же и для него  
пришел день, когда в кармане не осталось ни сантика. Не хватило даже  
денег, чтоб расплатиться с рабочими.  
Еще хорошо, что владелец приморских увеселительных заведений принял  
Пончика работать крутильщиком на чертовом колесе. Если бы не это, Пончик  
совсем остался бы без средств к существованию.

# Диаграмма классов реализованной объектной модели:



# Исходный код:

**Story.java**

**import** Exeptions.LunatikiExeption;  
**import** People.\*;  
**import** Quality.\*;  
  
  
**public class** Story {  
 **public static void** main(String[] args){  
 Ponchic ponchic = **new** Ponchic(**"Пончик"**, NationalWelfare.***Top10Forbes***, **false**,3);  
 Dracula dracula = **new** Dracula(**"Дракула"**, NationalWelfare.***Rich***, **false**, 4);  
 Lunatiki[] lunatik = **new** Lunatiki[9];  
 lunatik[0] = **new** Lunatiki(**"Лунатики"**, NationalWelfare.***MiddleClass***, 0);  
 lunatik[1] = **new** Lunatiki(**"Первый лунатик"**, NationalWelfare.***Poor***, 0);  
 lunatik[2] = **new** Lunatiki(**"Второй лунатик"**, NationalWelfare.***Poor***, 0);  
 lunatik[3] = **new** Lunatiki(**"Третий лунатик"**, NationalWelfare.***MiddleClass***, 0);  
 lunatik[4] = **new** Lunatiki(**"Четвертый лунатик"**, NationalWelfare.***MiddleClass***, 0);  
 lunatik[5] = **new** Lunatiki(**"Пятый лунатик"**, NationalWelfare.***MiddleClass***, 0);  
 lunatik[6] = **new** Lunatiki(**"Шестой лунатик"**, NationalWelfare.***MiddleClass***, 0);  
 lunatik[7] = **new** Lunatiki(**"Седьмой лунатик"**, NationalWelfare.***MiddleClass***, 0);  
 lunatik[8] = **new** Lunatiki(**"Восьмой лунатик"**, NationalWelfare.***Poor***, 0);  
 Factory factory = **new** Factory(**"Завод"**);  
 Factory.Salt salt = factory.**new** Salt(**"Соль"**);  
 Owner owner = **new** Owner(**"Владелец Приморского увеселительного заведения"**);  
 Owner.Amusement amusement = **new** Owner.Amusement();  
 Slaves slaves = **new** Slaves() {  
 @Override  
 **public void** dress() {  
 System.***out***.print(**"Они одевали "** + ponchic.getName() + **","**);  
 }  
 @Override  
 **public void** undress() {  
 System.***out***.println(**" они раздевали "**+ ponchic.getName() + **"."**);  
 }  
 @Override  
 **public void** clean() {  
 System.***out***.println(**"Они убирали у "**+ ponchic.getName() + **" в комнатах."**);  
 }  
 @Override  
 **public void** lookFor() {  
 System.***out***.println(**"Они смотрели за домом "**+ ponchic.getName() + **"."**);  
 }  
 };  
 ponchic.leave();  
 ponchic.newName();  
 ponchic.existHouse();  
 ponchic.slave();  
 slaves.dress();  
 slaves.undress();  
 slaves.clean();  
 slaves.lookFor();  
 ponchic.chill();  
 ponchic.condition();  
 ponchic.duration();  
  
 **if** (ponchic.getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**)) {  
 **for** (**int** i = 0; i< lunatik.**length**; i += 2){  
 lunatik[i].setKnowledge();  
 }  
 }  
 **for** (**int** i = 1; i < lunatik.**length**; i++){  
 **if** (lunatik[0].equals((lunatik[i]))){  
 lunatik[i].OpenAfactory();  
 lunatik[i].betterNationalWelfare();  
 salt.addiction();  
 ponchic.earnCh();  
 } **else** lunatik[i].setJob();  
 lunatik[i].NumWorker();  
  
 }  
 ponchic.worseNationalWelfare();  
 salt.printProductionVolume();  
 salt.printPrice();  
 salt.printStdIncome(factory.getNumber());  
 ponchic.getEarn(factory.getNumber());  
 salt.meaning(factory.getNumber());  
 **try** {  
 dracula.comeback(factory.getNumber());  
 } **catch** (LunatikiExeption e) {  
 e.printStackTrace();  
 }  
 dracula.setKnowledge();  
 **for** (**int** i = 1; i < lunatik.**length**; i++){  
 lunatik[i].steal();  
 }  
 salt.addiction();  
 dracula.fence();  
 ponchic.payDracula(dracula.getKnow());  
 ponchic.worseNationalWelfare();  
 **for** (**int** i = 1; i < lunatik.**length**; i++){  
 lunatik[i].payDracula(dracula.getKnow());  
 }  
 ponchic.taxes();  
 **for** (**int** i = 1; i < lunatik.**length**; i++){  
 lunatik[i].taxes();  
  
 }  
 salt.setIncome();  
 salt.printPrice();  
 ponchic.noProfit();  
  
 **for** (**int** i = 1; i < lunatik.**length**; i++) {  
 lunatik[i].noProfit();  
 }  
 **for** (**int** i = 1; i < lunatik.**length**; i++) {  
 lunatik[i].closeAFactory();  
 }  
 ponchic.condition();  
 ponchic.existHouse();  
 ponchic.closeAFactory();  
 ponchic.conditionCash();  
 owner.accept(ponchic.getName());  
 ponchic.worseNationalWelfare();  
 ponchic.condition();  
  
  
  
  
 }  
  
}

**Factory.java**

**package** Quality;  
  
**public class** Factory {  
 **public static int** *number*;  
 **private** String **name**;  
 **public** Factory(String name){  
 **this**.**name** = name;  
 }  
 **public int** getNumber(){  
 **return** *number*;  
 }  
 **public static void** addNumber(**int** num){  
 *number* +=num;  
 }  
 **public class** Salt {  
 **private** String **name**;  
 **private double productionVolume** = 28900;  
 **private double price** = 785;  
 **private static double** *income*;  
 **private final double priceStd** = 785;  
 **private final double productVolumeStd** = 28900;  
 **public** Salt(String name){  
 **this**.**name** = name;  
 }  
 **public void** addiction() {  
 **this**.**productionVolume** = **productionVolume** \* 1.2;  
 **this**.**price** = **price** \* 0.75;  
 }  
  
 **public void** printStdIncome(**int** num) {  
 **double** income = **price** \* **productionVolume** / num;  
 **if** (income > **priceStd** \* **productVolumeStd**) {  
 System.***out***.println(getName() + **" приносит больше барышей, чем вначале."**);  
 } **else** {  
 System.***out***.println(getName() + **" уже не приносит таких барышей как вначале."**);  
 }  
 }  
  
 **public void** meaning(**int** num) {  
 **int** min = 100000;  
 **if** (**price** \* **productionVolume** / num > min) {  
 System.***out***.println(**"Все еще выгодно, солевой промысел развивается"**);  
 } **else** {  
 System.***out***.println(**"Не выгодно, сворачиваемся"**);  
 }  
 }  
  
 **public void** printPrice() {  
 **if** (**price** > **priceStd**) {  
 System.***out***.println(getName() + **" продается все дороже."**);  
 } **else** {  
 **if** (**price** > 350) {  
 System.***out***.println(getName() + **" продается все дешевле."**);  
 }**else** {  
 System.***out***.println(getName() + **" стала продаваться по такой низкой цене, что "**);  
 }  
 }  
 }  
  
 **public void** printProductionVolume() {  
 **if** (**productionVolume** > **productVolumeStd**) {  
 System.***out***.println(getName() + **" добывается все больше."**);  
 } **else** {  
 System.***out***.println(getName() + **"Соли добывается все меньше."**);  
 }  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setIncome() {  
 *income* = **price** \* **productionVolume**;  
 }  
 **public static double** getIncome(){  
 **return** *income*;  
 }  
  
 }  
}

**NationalWeifware.java**

**package** Quality;  
  
**public enum** NationalWelfare{  
 ***Top10Forbes***(**"очень богат"**, **"огромная сумма"**, **"владеет всем"** ),  
 ***Rich***(**"богат"**, **"большая сумма"**, **"владеет всем побережьем"**),  
 ***MiddleClass***(**"средний класс"**, **"достаточная сумма"**, **"может владеть несколькими заводами"**),  
 ***Poor***(**"бедный"**, **"маленькая сумма"**, **"может владеть одним заводом"**),  
 ***Bum***(**"сводит концы с концами"**, **"в карманах ни сантика"**, **"все продал"**),  
 ***Nul***(**"без средств к существованию"**, **"пустота"**, **"ничего"**);  
  
 **public** String **title**;  
 **public** String **cash**;  
 **public** String **have**;  
  
 NationalWelfare(String title, String cash, String have){  
 **this**.**title** = title;  
 **this**.**cash** = cash;  
 **this**.**have** = have;  
 }  
}

**Protect.java**

**package** Quality;  
  
**public interface** Protect {  
 **void** fence();  
}

**Steal.java**

**package** Quality;  
  
**public interface** Steal {  
 **void** steal();  
  
}

**Slaves.java**

**package** Quality;  
  
**public interface** Slaves {  
 **void** undress();  
 **void** dress();  
 **void** clean();  
 **void** lookFor();  
  
}

**PersonAbstract.java**

**package** People;  
  
**import** Quality.Factory;  
**import** Quality.NationalWelfare;  
  
**public abstract class** PersonAbstract {  
 **private** String **name**;  
 **private** NationalWelfare **nationalWelfare**;  
 **private int have**;  
  
  
 **public** PersonAbstract(String name, NationalWelfare nationalWelfare, **int** have) {  
 **this**.**name** = name;  
 **this**.**nationalWelfare** = nationalWelfare;  
 **this**.**have** = have;  
 Factory.*addNumber*(have);  
 }  
  
 **public int** getHave(){  
 **return have**;  
 }  
 **public void** setHave(**int** have){  
 **this**.**have** = have;  
 }  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public** String getNationalWelfare() {  
 String title = **nationalWelfare**.**title**;  
 **return** title;  
 }  
 **public** String getNationalWelfwareCash(){  
 **return nationalWelfare**.**cash**;  
 }  
 **public** String getNationalWelfwareHave(){  
 String have = **nationalWelfare**.**have**;  
 **return** have;  
 }  
  
  
 **public void** betterNationalWelfare(){  
 System.***out***.println(getName() + **" приумножает свое богатство."**);  
 **if** (**nationalWelfare** == NationalWelfare.***Poor***){**nationalWelfare** = NationalWelfare.***MiddleClass***;}  
 **if** (**nationalWelfare** == NationalWelfare.***MiddleClass***){**nationalWelfare** = NationalWelfare.***Rich***;}  
 **if** (**nationalWelfare** == NationalWelfare.***Rich***){**nationalWelfare** = NationalWelfare.***Top10Forbes***;}  
 **if** (**nationalWelfare** == NationalWelfare.***Bum***){**nationalWelfare** = NationalWelfare.***Poor***;}  
 **if** (**nationalWelfare** == NationalWelfare.***Nul***){**nationalWelfare** = NationalWelfare.***Bum***;}  
 }  
 **public void** worseNationalWelfare(){  
 System.***out***.println(**"Дела "** + getName() + **"а идут хуже."**);  
 **if** (**nationalWelfare** == NationalWelfare.***Top10Forbes***){**nationalWelfare** = NationalWelfare.***Rich***;}  
 **if** (**nationalWelfare** == NationalWelfare.***Rich***){**nationalWelfare** = NationalWelfare.***MiddleClass***;}  
 **if** (**nationalWelfare** == NationalWelfare.***MiddleClass***){**nationalWelfare** = NationalWelfare.***Poor***;}  
 **if** (**nationalWelfare** == NationalWelfare.***Poor***){**nationalWelfare** = NationalWelfare.***Bum***;}  
 **if** (**nationalWelfare** == NationalWelfare.***Bum***){**nationalWelfare** = NationalWelfare.***Nul***;}  
 }  
  
 **public void** condition(){  
 System.***out***.println(getName()+ **" "**+getNationalWelfare() + **"."**);  
 }  
 **public void** conditionCash(){  
 System.***out***.println(**"Для "** + getName()+ **" пришел день, когда "** + getNationalWelfwareCash() + **"."**);  
 }  
 **public void** payDracula(**boolean** n){  
 }  
 **public void** taxes(){  
 }  
 **public abstract double** getProfit();  
 **public abstract void** OpenAfactory();  
 **public abstract void** closeAFactory();  
  
  
}

**Dracula.java**

**package** People;  
  
**import** Exeptions.LunatikiExeption;  
**import** Quality.Factory;  
**import** Quality.NationalWelfare;  
**import** Quality.Protect;  
  
**public class** Dracula **extends** PersonAbstract **implements** Protect {  
 **private boolean know**;  
 **public** Dracula(String name, NationalWelfare nationalWelfare, **boolean** know, **int** have) {  
 **super**(name, nationalWelfare, have);  
 **this**.**know** = know;  
 }  
 **public void** comeback(**int** a) **throws** LunatikiExeption{  
 **if** (a>0) {  
 System.***out***.println(**"Вернулся из поездки "** + getName() + **" который "** + getNationalWelfwareHave() + **"."**);  
 } **else** {  
 **throw new** LunatikiExeption(**"Лунатики еще ничего не построили."**);  
 }  
  
 }  
  
 **public void** setKnowledge(){  
 **know** = **true**;  
 System.***out***.print(getName() + **" узнал, что "**);  
 }  
 **public boolean** getKnow(){  
 **return know**;  
 }  
  
 @Override  
 **public double** getProfit() {  
 **double** personalIncom = Factory.Salt.*getIncome*() \* getHave() - (**double**) Lunatiki.*getNumWorker*()/getHave();  
 **return** personalIncom;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Имя:"** + getName() + **". Состояние: "** + getNationalWelfare()+ **". Знает о воровстве: "** + getKnow() + **". "**;  
 }  
 @Override  
 **public void** fence() {  
 **if** (**know**){  
 System.***out***.println(getName() + **" велит обнести побережье забором."**);  
 } **else** {  
 System.***out***.println(**" не замечает воровства."**);  
 }  
 }  
  
 @Override  
 **public void** OpenAfactory() {  
 **this**.setHave(getHave()+1);  
 System.***out***.println(getName() + **" открыл еще один завод."**);  
 }  
  
 @Override  
 **public void** closeAFactory() {  
 **this**.setHave(getHave()-1);  
 System.***out***.println(getName() + **"закрыл завод."**);  
 }  
}

**Lunatiki.java**

**package** People;  
  
**import** Quality.Factory;  
**import** Quality.NationalWelfare;  
**import** Quality.Steal;  
  
**public class** Lunatiki **extends** PersonAbstract **implements** Steal {  
 **private boolean know**;  
 **private boolean worker**;  
 **private static int** *numWorker*;  
 **private boolean taxes**;  
  
 **public** Lunatiki(String name, NationalWelfare nationalWelfare, **int** have) {  
 **super**(name, nationalWelfare, have);  
 }  
 **public void** setKnowledge(){  
 **know** = **true**;  
 System.***out***.println(getName() + **" теперь знает где брать соль. "**);  
  
 }  
 **public boolean** getKnowledge(){  
 **return know**;  
 }  
 **public void** setJob(){  
 **this**.**worker** = **true**;  
 }  
 **public boolean** getWorker(){  
 **return worker**;  
 }  
 **public void** NumWorker(){  
 **if** (getWorker()){  
 **this**.*numWorker* += 1;  
 }  
 }  
 **public static int** getNumWorker(){  
 **return** *numWorker*;  
 }  
  
 **public void** noProfit(){  
 **if** (!**worker**) {  
 **if** (**taxes**) {  
 System.***out***.println(getName() + **" приходится расходовать на приобретение соляных кристаллов и оплату рабочих гораздо больше денег,чем\n"** +  
 **"он выручал от продажи своей продукции. "**);  
 }  
 }  
 }  
  
 @Override  
 **public double** getProfit() {  
 **double** personalIncom = Factory.Salt.*getIncome*() \* getHave();  
 **return** personalIncom;  
 }  
  
 @Override  
 **public void** OpenAfactory() {  
 System.***out***.println(getName() + **" сберег "** + getNationalWelfwareCash() + **" и строит завод."**);  
 setHave(1);  
 Factory.*number*++;  
 Factory.*addNumber*(getHave());  
 System.***out***.println(**"Появляется небольшой соляной заводик. "**);  
 }  
  
 @Override  
 **public void** closeAFactory() {  
 **if** (!**worker**) {  
 System.***out***.println(getName() + **" закрывает свой соляной заводик."**);  
 setHave(Factory.*number*--);  
 }  
 }  
  
 @Override  
 **public void** steal() {  
 **if** (getHave() != 0){  
 System.***out***.println(getName() + **" ворует кристаллы для переработки на соляном заводе. "**);  
 }  
 }  
  
 @Override  
 **public void** payDracula(**boolean** n) {  
 **if** (n && getHave()!= 0){  
 System.***out***.println(getName() + **" платит Дракуле за кристаллы. "**);  
 }  
 }  
  
 @Override  
 **public void** taxes() {  
 **if** (getHave() != 0){  
 System.***out***.println(getName() + **" платит половину прибыли с завода в карман Дракулы. "**);  
 worseNationalWelfare();  
 **taxes** = **true**;  
 }  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Имя:"** + getName() + **". Состояние: "** + getNationalWelfare()+ **". Знает где брать соль: "** + getKnowledge() + **". "**;  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return** getNationalWelfare().hashCode() + Boolean.*toString*(getKnowledge()).hashCode();  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (o **instanceof** Lunatiki){  
 **if** (**this**.hashCode() == o.hashCode()) {  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
}

**Ponchic.java**

**package** People;  
  
**import** Errors.MessageError;  
**import** Quality.\*;  
  
**public class** Ponchic **extends** PersonAbstract{  
 **private boolean duration**;  
 **private final double earnStd** = 20;  
 **private double earn** = 20;  
 **private** String **newName** = **"Господин Понч"**;  
 **private boolean slave**;  
 **private boolean taxes**;  
 **public** Ponchic(String name, NationalWelfare nationalWelfare, **boolean** duration, **int** have) {  
 **super**(name, nationalWelfare, have);  
 **this**.**duration** = duration;  
  
 }  
 *//Локальный класс* **public void** existHouse(){  
 **boolean** welfware = getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**);  
 String ponchicname = getName();  
 **class** House{  
 **private** String **name**;  
 **public** House(String name){  
 **this**.**name** = name;  
 }  
 **public** String getName() {  
 **return name**;  
 }  
 **public void** exist(){  
 **if** (welfware) {  
 System.***out***.println(ponchicname + **" переехал в "** + getName() + **"."**);  
 } **else** {  
 System.***out***.println(ponchicname + **" продал свой "** + getName() + **"."**);  
 }  
 }  
 }  
 House house = **new** House(**"Собственный дом"**);  
 house.exist();  
 }  
 **public void** noProfit(){  
 **if**(**taxes**) {  
 System.***out***.println(getName() + **" приходится расходовать на приобретение соляных кристаллов и оплату рабочих гораздо больше денег,чем\n"** +  
 **"он выручал от продажи своей продукции. "**);  
 }  
 }  
  
 **public void** chill(){  
 **if** (getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**) && **slave**){  
 System.***out***.println(getName() + **" целыми днями просиживал в ресторанах, ел там самые вкусные кушанья."**);  
 System.***out***.println(getName() + **"Околачивался на берегу залива. "**);  
 Owner.Amusement.*spinner*(getName());  
 Owner.Amusement.*sea*(getName());  
  
 } **else** {  
 System.***out***.println(getName() + **" работал."**);  
 }  
 }  
 **public void** setSlave(){  
 **this**.**slave** = **true**;  
 }  
 **public void** slave(){  
 **if** (getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**)){  
 System.***out***.println(getName() + **" завел себе слуг."**);  
 setSlave();  
 } **else** {  
 System.***out***.println(getName() + **" делает все сам."**);  
 }  
 }  
 **public void** leave(){  
 **if** (getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**)){  
 System.***out***.println(**"Жил "** + getName() + **" в свое удовольствие, как и остальные лунные богачи."**);  
 } **else** {  
 System.***out***.println(**"Жил "** + getName() + **" скудно"**);  
 }  
 }  
 **public void** newName(){  
 **if** (!getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**)){  
 System.***out***.println(**"Называется как обычно - "** + getName());  
 } **else** {  
 System.***out***.println(**"И даже назывался теперь не просто Пончик, a "** + getName() + **"."**);  
 }  
 }  
  
 **public void** duration(){  
 **if** (**duration**) {  
 System.***out***.println(**" долго. "**);  
 } **else** {  
 System.***out***.println(**" недолго. "**);  
 }  
  
 }  
 **public void** earnCh(){  
 **this**.**earn** = **earn** - 9.5;  
 }  
  
 **public void** getEarn(**int** n) **throws** MessageError{  
 **if** (n < 20) {  
 System.***out***.println(getName() + **" раньше, тратит на оплату рабочего фертинг и получает взамен "** + **earnStd** + **"."**);  
 **if** (**earn** > **earnStd**) {  
 System.***out***.println(**"Теперь рабочий приносит целых"** + **earn** + **"."**);  
 } **else** {  
 System.***out***.println(**"Теперь никто не может выколотить из рабочего и "** + **earn** + **" фертинга."**);  
 }  
 } **else** {  
 **throw new** MessageError(**"Слишком много заводов!"**);  
 }  
  
 }  
  
 @Override  
 **public** String getName() {  
 **if** (!getNationalWelfare().equals(NationalWelfare.***Top10Forbes***.**title**)) {  
 **return super**.getName();  
 }**else** {  
 **return newName**;  
 }  
 }  
  
 @Override  
 **public void** condition() {  
 **super**.condition();  
 }  
  
 @Override  
 **public double** getProfit() {  
 **double** personalIncom = Factory.Salt.*getIncome*() \* getHave()-(**double**) Lunatiki.*getNumWorker*()/getHave();  
 **return** personalIncom;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Имя:"** + getName() + **". Состояние: "** + getNationalWelfare()+ **". "**;  
 }  
 @Override  
 **public void** payDracula(**boolean** n){  
 **if** (n) {  
 System.***out***.println(getName() + **" платит Дракуле за кристаллы."**);  
 } **else** {System.***out***.println(getName() + **" ура налогов нет."**);}  
 }  
  
 @Override  
 **public void** taxes() {  
 System.***out***.println(getName() + **" платит половину прибыли с завода в карман Дракулы. "**);  
 worseNationalWelfare();  
 **taxes** = **true**;  
 }  
  
 @Override  
 **public void** OpenAfactory() {  
 **this**.setHave(getHave()+1);  
 System.***out***.println(getName() + **" открыл еще один завод."**);  
 }  
  
 @Override  
 **public void** closeAFactory() {  
 **this**.setHave(getHave()-1);  
 **for** (**int** i = 1; i <= getHave() + 1; i++)  
 System.***out***.println(getName() + **" закрыл свой "** + i + **" завод."**);  
 }  
  
}

**Owner.java**

**package** People;  
  
**public class** Owner {  
 **private** String **name**;  
 **public** Owner(String name){  
 **this**.**name** = name;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** accept(String name){  
 System.***out***.println(getName() + **" Принял "** + name + **"а работать крутильщиком чертового колеса. "**);  
 System.***out***.println(**"Иначе"**);  
 }  
 **public static class** Amusement {  
  
 **public static void** spinner(String name) {  
 System.***out***.println(**"Крутит "** + name + **" на чертовых водяных колесах."**);  
 }  
  
 **public static void** sea(String name) {  
 System.***out***.println(**"Крутит "** + name + **" на морском параболоиде."**);  
 }  
 }  
  
}

**LunatikiExeption.java**

**package** Exeptions;  
  
**public class** LunatikiExeption **extends** Exception {  
 **public** LunatikiExeption(String message){  
 **super**(message);  
  
 }  
}

**MessageError.java**

**package** Errors;  
  
**public class** MessageError **extends** Error{  
 **public** MessageError(String message){  
 **super**(message);  
 }  
}

# Результаты работы программы:

Жил Господин Понч в свое удовольствие, как и остальные лунные богачи.

И даже назывался теперь не просто Пончик, a Господин Понч.

Господин Понч переехал в Собственный дом.

Господин Понч завел себе слуг.

Они одевали Господин Понч, они раздевали Господин Понч.

Они убирали у Господин Понч в комнатах.

Они смотрели за домом Господин Понч.

Господин Понч целыми днями просиживал в ресторанах, ел там самые вкусные кушанья.

Господин ПончОколачивался на берегу залива.

Крутит Господин Понч на чертовых водяных колесах.

Крутит Господин Понч на морском параболоиде.

Господин Понч очень богат.

недолго.

Лунатики теперь знает где брать соль.

Второй лунатик теперь знает где брать соль.

Четвертый лунатик теперь знает где брать соль.

Шестой лунатик теперь знает где брать соль.

Восьмой лунатик теперь знает где брать соль.

Четвертый лунатик сберег достаточная сумма и строит завод.

Появляется небольшой соляной заводик.

Четвертый лунатик приумножает свое богатство.

Шестой лунатик сберег достаточная сумма и строит завод.

Появляется небольшой соляной заводик.

Шестой лунатик приумножает свое богатство.

Дела Господин Понча идут хуже.

Соль добывается все больше.

Соль продается все дешевле.

Соль уже не приносит таких барышей как вначале.

Пончик раньше, тратит на оплату рабочего фертинг и получает взамен 20.0.

Теперь никто не может выколотить из рабочего и 1.0 фертинга.

Все еще выгодно, солевой промысел развивается

Вернулся из поездки Дракула который владеет всем побережьем.

Дракула узнал, что Четвертый лунатик ворует кристаллы для переработки на соляном заводе.

Шестой лунатик ворует кристаллы для переработки на соляном заводе.

Дракула велит обнести побережье забором.

Пончик платит Дракуле за кристаллы.

Дела Пончика идут хуже.

Четвертый лунатик платит Дракуле за кристаллы.

Шестой лунатик платит Дракуле за кристаллы.

Пончик платит половину прибыли с завода в карман Дракулы.

Дела Пончика идут хуже.

Четвертый лунатик платит половину прибыли с завода в карман Дракулы.

Дела Четвертый лунатика идут хуже.

Шестой лунатик платит половину прибыли с завода в карман Дракулы.

Дела Шестой лунатика идут хуже.

Соль стала продаваться по такой низкой цене, что

Пончик приходится расходовать на приобретение соляных кристаллов и оплату рабочих гораздо больше денег,чем

он выручал от продажи своей продукции.

Четвертый лунатик приходится расходовать на приобретение соляных кристаллов и оплату рабочих гораздо больше денег,чем

он выручал от продажи своей продукции.

Шестой лунатик приходится расходовать на приобретение соляных кристаллов и оплату рабочих гораздо больше денег,чем

он выручал от продажи своей продукции.

Четвертый лунатик закрывает свой соляной заводик.

Шестой лунатик закрывает свой соляной заводик.

Пончик без средств к существованию.

Пончик продал свой Собственный дом.

Пончик закрыл свой 1 завод.

Пончик закрыл свой 2 завод.

Пончик закрыл свой 3 завод.

Для Пончик пришел день, когда пустота.

Владелец Приморского увеселительного заведения Принял Пончика работать крутильщиком чертового колеса.

Иначе

Дела Пончика идут хуже.

Пончик без средств к существованию.

# Вывод:

В ходе лабораторной работы я закрепила основные принципы ООП, принципы SOLID и изучила вложенные, локальные и анонимные классы.