Санкт-Петербургский Национальный Исследовательский

Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №4**

По дисциплине «Программирование»

**Вариант 23014**

Выполнила: Мозговая Лариса Андреевна

Группа P3130

Преподаватель: Блохина Елена Николаевна

Санкт-Петербург

2022

**Задание:**

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

Получив эту телеграмму, Крабс понял, что Спрутс решил дать Миге и Жулио не два, а лишь один миллион. Это нисколько не удивило Крабса, так как он хорошо знал, что господин Спрутс действует всегда осмотрительно и на ветер денег бросать не станет. Крабса удовлетворяло то, что Спрутс не отказался уплатить деньги, и теперь можно было надеяться, что, согласившись распроститься с одним миллионом, он под конец расстанется и с двумя.  
Всесторонне обдумав создавшееся положение, господин Крабс решил ничего не говорить о полученной телеграмме Миге и Жулио, так как, узнав её содержание, они тоже пришли бы к мысли, что дела складываются, в общем, успешно, и могли бы ещё больше повысить цену за своё исчезновение. Встретившись с ними, господин Крабс сказал, что никаких известий от господина Спрутса нет, но надежды на успешное завершение дела терять не следует.  
Его заявление все же опечалило господина Жулио, которому не терпелось поскорее удрать со всеми деньгами.  
В действительности Крабс никому не стал в этот день телеграфировать. Вместо этого он пошёл в ресторан и сытно пообедал. Потом вернулся к себе в гостиницу, всхрапнул часок, потом искупался в плавательном бассейне, после чего снова встретился с Мигой и Жулио. Собравшись втроём, друзья сначала поужинали, а потом отправились в ночной театр, где за небольшую плату разрешалось швырять в актёров гнилыми яблоками, и как следует повеселились.  
Проснувшись на следующий день, господин Крабс никому не сказал ни слова и отправил Спрутсу телеграмму.  
В ответ от Спрутса в тот же день была получена телеграмма.  
Получив эту телеграмму, Крабс выждал ещё денёк и, ничего не сказав ни Миге, ни Жулио, отправил Спрутсу еще одну телеграмму.  
Неизвестно, до чего бы дошёл этот обмен телеграммами, если бы на следующее утро в гостинице, где остановился господин Крабс, появился вдруг Скуперфильд со своей палкой в руках и в обычном своём наряде, который состоял из чёрного длиннополого пиджака с двумя разрезами на спине, чёрных брюк и чёрной высокой шляпы, известной под названием цилиндра. Столкнувшись с Крабсом, который как раз в этот момент выходил из гостиницы, Скуперфильд раскрыл широко объятия и завизжал своим отвратительным голосом.

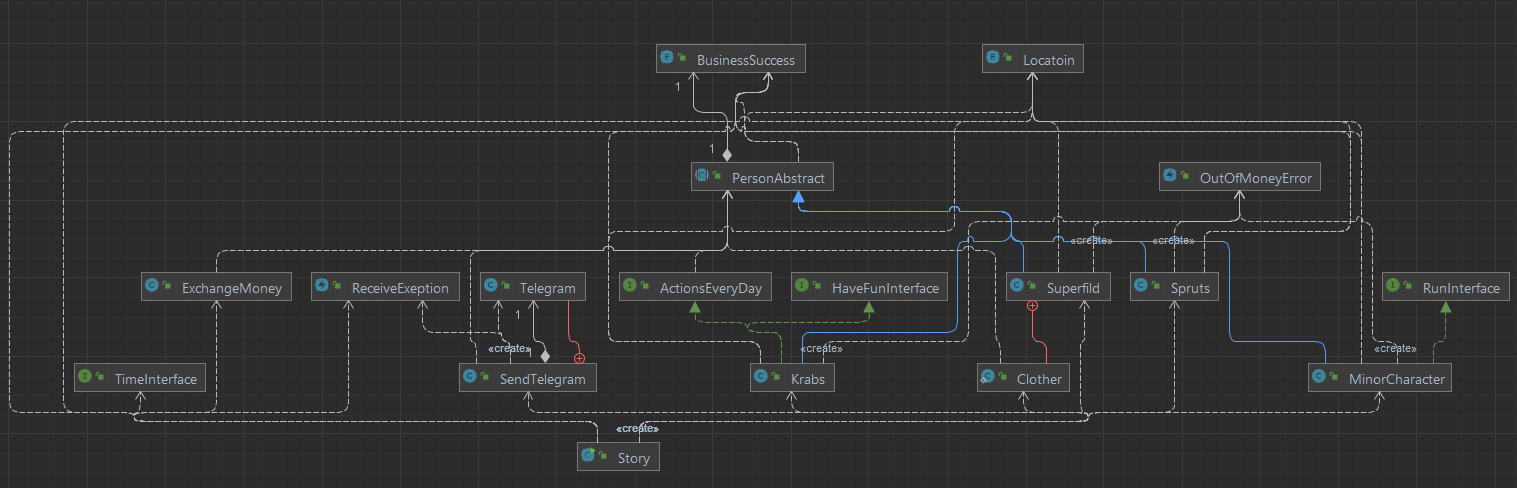
**Программа должна удовлетворять следующим требованиям:**

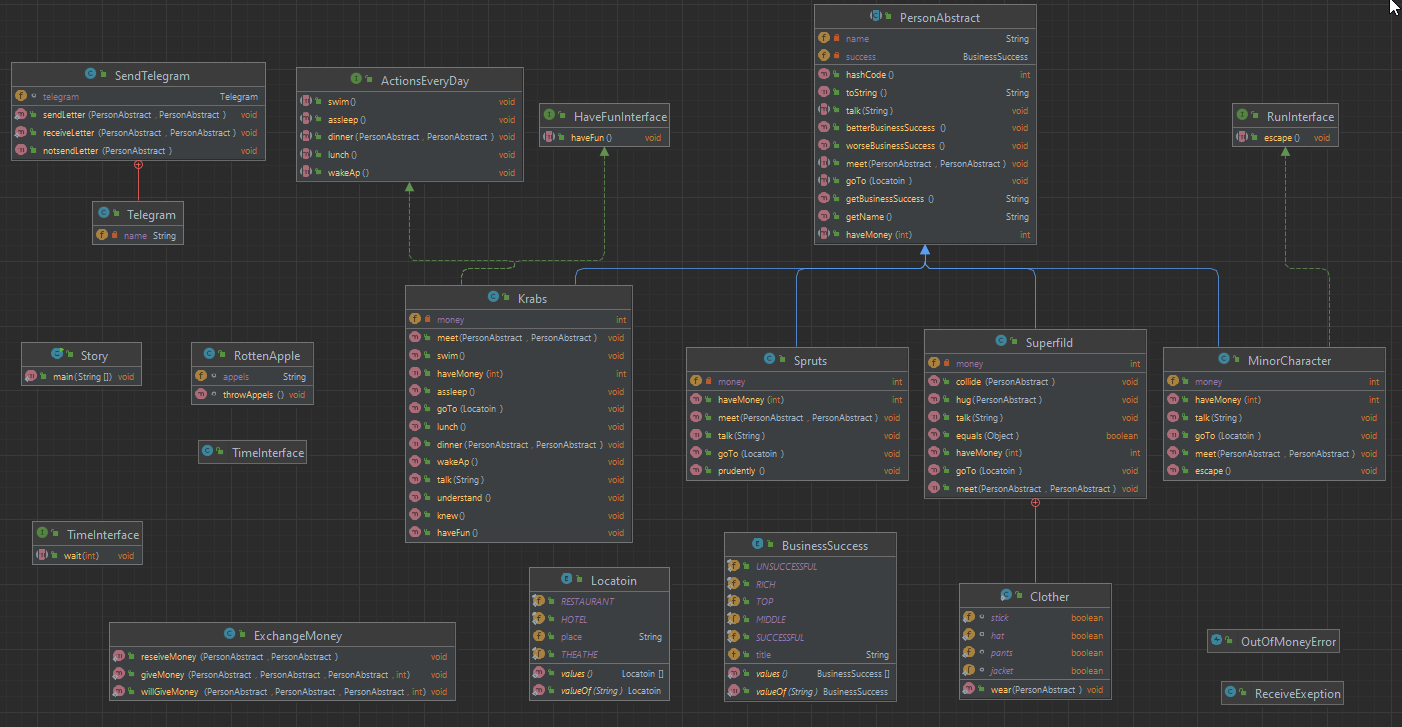
1. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
2. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

**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 UML:**

****

****

**Код программы:**

**Story.java**

import utility.Locatoin;  
import utility.TimeInterface;  
  
public class Story {  
 public static void main(String[] args) {  
 Krabs krabs = new Krabs("Крабс", BusinessSuccess.*SUCCESSFUL*);  
 Spruts spruts = new Spruts("Спрутс", BusinessSuccess.*TOP*);  
 MinorCharacter julio = new MinorCharacter("Жулио", BusinessSuccess.*UNSUCCESSFUL*);  
 MinorCharacter migga = new MinorCharacter("Мигга", BusinessSuccess.*UNSUCCESSFUL*);  
 Superfild superfild = new Superfild("Суперфильд");  
 SendTelegram sendTelegram = new SendTelegram();  
 Superfild.Clother clother = new Superfild.Clother();  
  
  
 System.*out*.println(spruts);  
 System.*out*.println(migga);  
 System.*out*.println(julio);  
  
 try {  
 SendTelegram.*sendLetter*(spruts,krabs);  
 } catch (ReceiveExeption e) {  
 e.printStackTrace();  
 }  
 ExchangeMoney.*giveMoney*(spruts, migga, julio, 1000000000);  
 krabs.knew();  
 spruts.prudently();  
 krabs.understand();  
 ExchangeMoney.*willGiveMoney*(spruts, migga, julio, 2000000);  
 krabs.talk("Мигге и Жулио");  
 krabs.meet(migga,julio);  
  
 julio.escape();  
  
 sendTelegram.notsendLetter(krabs);  
 krabs.goTo(Locatoin.*RESTAURANT*);  
 krabs.goTo(Locatoin.*HOTEL*);  
 krabs.meet(migga,julio);  
 krabs.dinner(migga,julio);  
 migga.goTo(Locatoin.*THEATHE*);  
 julio.goTo(Locatoin.*THEATHE*);  
 krabs.goTo(Locatoin.*THEATHE*);  
  
 krabs.wakeAp();  
 krabs.talk("никому");  
 try {  
 SendTelegram.*sendLetter*(krabs,spruts);  
 } catch (ReceiveExeption e) {  
 e.printStackTrace();  
 }  
 try {  
 SendTelegram.*sendLetter*(spruts,krabs);  
 } catch (ReceiveExeption e) {  
 e.printStackTrace();  
 }  
 TimeInterface time = new TimeInterface() {  
 @Override  
 public void wait(int time) {  
 System.*out*.println("ожидание " + time + " день");  
 }  
 };  
 time.wait(1);  
 krabs.talk("Мигга и Жулио");  
  
 try {  
 SendTelegram.*sendLetter*(krabs,spruts);  
 } catch (ReceiveExeption e) {  
 e.printStackTrace();  
 }  
 superfild.goTo(Locatoin.*HOTEL*);  
 clother.*wear*(superfild);  
 superfild.collide(krabs);  
 superfild.hug(krabs);  
 superfild.talk("-");  
  
 }  
}

**ActionsEveryDay.java**

package utility;  
  
public interface ActionsEveryDay {  
 void assleep();  
 void swim();  
 void dinner(PersonAbstract person, PersonAbstract person2);  
 void lunch();  
 void wakeAp();  
  
}

**HaveFunInterface.java**

package utility;  
  
public interface HaveFunInterface {  
 void haveFun();  
}

**RunInterface.java**

package utility;  
  
public interface RunInterface {  
 void escape();  
}

**TimeInterface.java**

package utility;  
  
public interface TimeInterface {  
 void wait(int time);  
}

**BusinessSuccess.java**

package utility;  
  
public enum BusinessSuccess {  
 *UNSUCCESSFUL*("неуспешный"),  
 *MIDDLE*("среднеуспешный"),  
 *SUCCESSFUL*("успешный"),  
 *RICH*("очень успешный"),  
 *TOP*("самый успешный");  
  
 public String title;  
  
 BusinessSuccess(String title){  
 this.title = title;  
 }  
}

**Locatoin.java**

package utility;  
  
public enum Locatoin {  
 *RESTAURANT*("ресторан"),  
 *HOTEL*("гостиница"),  
 *THEATHE*("театр");  
  
 public String place;  
  
 Locatoin(String place){  
 this.place = place;  
 }  
}

**PersonAbstract.java**

package utility;  
  
public abstract class PersonAbstract {  
 private String name;  
 private BusinessSuccess success;  
 public PersonAbstract(String name, BusinessSuccess success){  
 this.name = name;  
 this.success = success;  
 }  
 public PersonAbstract(String name){  
 this.name = name;  
 }  
 public String getName(){  
 return this.name;  
 }  
 public String getBusinessSuccess() {  
 String title = success.title;  
 return title;  
 }  
 public void betterBusinessSuccess(){  
 if(success == BusinessSuccess.*RICH*){  
 success = BusinessSuccess.*TOP*;  
 }  
 if(success == BusinessSuccess.*SUCCESSFUL*){  
 success = BusinessSuccess.*RICH*;  
 }  
 if(success == BusinessSuccess.*MIDDLE*){  
 success = BusinessSuccess.*SUCCESSFUL*;  
 }  
 if(success == BusinessSuccess.*UNSUCCESSFUL*){  
 success = BusinessSuccess.*MIDDLE*;  
 }  
 System.*out*.println("Бизнесс " + getName()+ " стал более успешный");  
 }  
  
 public void worseBusinessSuccess(){  
 if(success == BusinessSuccess.*MIDDLE*){  
 success = BusinessSuccess.*UNSUCCESSFUL*;  
 }  
 if(success == BusinessSuccess.*SUCCESSFUL*){  
 success = BusinessSuccess.*MIDDLE*;  
 }  
 if(success == BusinessSuccess.*RICH*){  
 success = BusinessSuccess.*SUCCESSFUL*;  
 }  
 if(success == BusinessSuccess.*TOP*){  
 success = BusinessSuccess.*RICH*;  
 }  
 System.*out*.println("Бизнесс " + getName()+ " стал менее успешный");  
 }  
 public abstract void talk(String people);  
 public abstract void meet(PersonAbstract person1,PersonAbstract person2);  
 public abstract void goTo(Locatoin locatoin);  
 public abstract int haveMoney(int profit);  
  
 @Override  
 public String toString() {  
 return "Имя: " + getName() + ". Бизнесс: " + getBusinessSuccess();  
 }  
 @Override  
 public int hashCode(){  
 int result = 0;  
 result+= 31 \* name.hashCode();  
 result+= 32 \* success.hashCode();  
 return result;  
 }  
}

**OutOfMoneyError.java**

package errors;  
  
public class OutOfMoneyError extends Error {  
 public OutOfMoneyError(String message) {  
 super(message);  
 }  
}

ReceiveExeption.java

package exeption;  
  
public class ReceiveExeption extends Exception {  
 public ReceiveExeption(String message){  
 super(message);  
 }  
}

**ExchangeMoney.java**

package model;  
  
import utility.PersonAbstract;  
  
public class ExchangeMoney {  
 private ExchangeMoney() {}  
 public static void reseiveMoney(PersonAbstract recipiend,PersonAbstract recipiend1) {  
 recipiend1.betterBusinessSuccess();  
 recipiend.betterBusinessSuccess();  
 }  
 public static void giveMoney(PersonAbstract giver, PersonAbstract recipiend, PersonAbstract recipiend1, int profit) {  
 System.*out*.println(giver.getName() + " дал " + recipiend.getName() + " и " + recipiend1.getName() + " " + profit);  
 ExchangeMoney.*reseiveMoney*(recipiend,recipiend1);  
 giver.worseBusinessSuccess();  
 giver.haveMoney(-profit);  
 recipiend.haveMoney(profit/2);  
 recipiend1.haveMoney(profit/2);  
 }  
 public static void willGiveMoney(PersonAbstract giver, PersonAbstract recipiend, PersonAbstract recipiend1, int profit) {  
 System.*out*.println(giver.getName() + " даст " + recipiend.getName() + " и " + recipiend1.getName() + " " + profit);  
 }  
}

**Krabs.java**

package model;  
  
import errors.OutOfMoneyError;  
import utility.\*;  
  
public class Krabs extends PersonAbstract implements ActionsEveryDay,HaveFunInterface{  
 private int money = 79000000;  
 public Krabs(String name,BusinessSuccess success){  
 super(name, success);  
 }  
  
 @Override  
 public void assleep() {  
 System.*out*.printf(getName()+ " поспал%n");  
 }  
  
 @Override  
 public void wakeAp() {  
 System.*out*.println(getName()+ " проснулся");  
 }  
  
 @Override  
 public void swim() {  
 System.*out*.printf(getName()+ " искупался в бассейне%n%n");  
 }  
 @Override  
 public int haveMoney(int profit) throws OutOfMoneyError {  
 money += profit;  
 if (money >= 0){  
 return money;  
 }  
 else {  
 throw new OutOfMoneyError(getName() + " разорился. Деньги закончились!");  
 }  
 }  
  
 @Override  
 public void dinner(PersonAbstract person, PersonAbstract person2) {  
 if (person.getBusinessSuccess().equals(person2.getBusinessSuccess())){  
 System.*out*.println(getName() + " поужинал с друзьями");  
 }  
 else{  
 System.*out*.println(getName() + " поужинал с " + person.getName() + " и " + person2.getName());  
 }  
 }  
  
 @Override  
 public void lunch() {  
 System.*out*.printf(getName() + " пообедал%n%n");  
 }  
  
 @Override  
 public void meet(PersonAbstract person1,PersonAbstract person2) {  
 System.*out*.println(getName() + " встретился с " + person1.getName() + " и " + person2.getName());  
 }  
  
 @Override  
 public void talk(String people) {  
 System.*out*.println(getName() + " ничего не сказал " + people);  
 }  
  
  
 @Override  
 public void goTo(Locatoin locatoin) {  
 if (locatoin == Locatoin.*RESTAURANT*) {  
 System.*out*.println(getName() + " пошёл в " + locatoin.place + ": ");  
 lunch();  
 }  
 if (locatoin == Locatoin.*HOTEL*) {  
 System.*out*.println(getName() + " пошёл в " + locatoin.place + ": ");  
 assleep();  
 swim();  
 }  
 if (locatoin == Locatoin.*THEATHE*) {  
 System.*out*.println(getName() + " пошёл в " + locatoin.place + ": ");  
 haveFun();  
 }  
  
 }  
  
 public void understand(){  
 System.*out*.print(getName()+ " надеется: ");  
 };  
 public void knew(){  
 System.*out*.print(getName()+ " знал: ");  
 };  
  
 @Override  
 public void haveFun() {  
 class RottenApple{  
 String appels = "гнилые яблоки";  
 void throwAppels(){  
 System.*out*.println("В театре можно швырять в актёров "+ appels + ". ");  
 }  
 }  
 RottenApple rottenApple = new RottenApple();  
 rottenApple.throwAppels();  
 System.*out*.printf("Можно веселиться.%n%n");  
 }  
}

**MinorCharacter.java**

import com.ifmo.programming.utility.\*;  
  
public class MinorCharacter extends PersonAbstract implements RunInterface {  
 public int money;  
 public MinorCharacter(String name, BusinessSuccess success){  
 super(name, success);  
 }  
 @Override  
 public int haveMoney(int profit) throws OutOfMoneyError {  
 money += profit;  
 if (money < 0){  
 throw new OutOfMoneyError(getName() + "разорился. Деньги закончились! ");  
 }  
 return money;  
 }  
  
 @Override  
 public void meet(PersonAbstract person1,PersonAbstract person2) {  
 System.*out*.println(getName() + " встретился с " + person1 + " и " + person2);  
 }  
  
 @Override  
 public void talk(String people) {  
 System.*out*.println(getName() + " ничего не сказал " + people);  
 }  
  
  
 @Override  
 public void goTo(Locatoin locatoin) {  
 if (locatoin == Locatoin.*THEATHE*) {  
 System.*out*.println(getName() + " пошёл в " + locatoin.place + ": ");  
 }  
 }  
  
 @Override  
 public void escape() {  
 if (getBusinessSuccess() != BusinessSuccess.*UNSUCCESSFUL*.title) {  
 System.*out*.println(getName() + " хочет удрать со всеми деньгами");  
 }  
 }  
}

**SendTelegram.java**

package model;  
  
import exeption.ReceiveExeption;  
import utility.PersonAbstract;  
  
public class SendTelegram {  
 public SendTelegram() {  
 }  
  
 public static void receiveLetter(PersonAbstract sender, PersonAbstract recipiend) throws ReceiveExeption {  
 if (Math.*random*() < 0.3){  
 throw new ReceiveExeption("Телеграмма не была доставлена "+ recipiend.getName());  
 }  
 else {  
 System.*out*.printf(recipiend.getName() + " получил телеграмму от " + sender.getName()+"%n%n");  
 }  
 }  
  
 public static void sendLetter(PersonAbstract sender, PersonAbstract recipiend) throws ReceiveExeption {  
 System.*out*.println(sender.getName() + " отправил телеграмму " + recipiend.getName() + " -> ");  
 SendTelegram.*receiveLetter*(sender, recipiend);  
  
 }  
 public void notsendLetter(PersonAbstract sender){  
 System.*out*.println(sender.getName()+" решил не отправлять " + telegram.name);  
 }  
 Telegram telegram = new Telegram("телеграмма");  
 public class Telegram {  
 private String name;  
 public Telegram(String name) {  
 this.name = name;  
 }  
  
 }  
}

**Spruts.java**

package model;  
  
import errors.OutOfMoneyError;  
import utility.\*;  
  
public class Spruts extends PersonAbstract{  
 private int money = 100000000;  
 public Spruts(String name,BusinessSuccess success) {  
 super(name, success);  
 }  
 public void prudently(){  
 System.*out*.println(getName()+ " действует разумно и осмотрительно");  
 }  
  
 @Override  
 public int haveMoney(int profit) throws OutOfMoneyError {  
 money += profit;  
 if (money >= 0){  
 return money;  
 }  
 else {  
 throw new OutOfMoneyError(getName() + " разорился. Деньги закончились!");  
 }  
 }  
  
 @Override  
 public void meet(PersonAbstract person1,PersonAbstract person2) {  
 System.*out*.println(getName() + " встретился с " + person1 + " и " + person2);  
 }  
  
 @Override  
 public void talk(String people) {  
 System.*out*.println(getName() + " что-то сказал " + people);  
 }  
  
  
 @Override  
 public void goTo(Locatoin locatoin ) {  
 System.*out*.println(getName() + " пошел в " + locatoin.place);  
 }  
  
}

**Superfild.java**

package model;  
  
import errors.OutOfMoneyError;  
import utility.Locatoin;  
import utility.PersonAbstract;  
  
public class Superfild extends PersonAbstract {  
 private int money = 5000000;  
 public Superfild(String name){  
 super(name);}  
 public void hug(PersonAbstract person){  
 if (getName().equals(person.getName())){  
 System.*out*.println(getName() + " обнял сам себя");  
 }  
 else {  
 System.*out*.println(getName()+ " обнял " + person.getName());  
 }  
 }  
 public void collide(PersonAbstract person){  
 if (getName().equals(person.getName())){  
 System.*out*.println(getName() + " не может сталкнуться с самим собой!");  
 }  
 else {  
 System.*out*.println(getName() + " столкнулся с " + person.getName());  
 }  
 }  
  
 @Override  
 public void meet(PersonAbstract person1,PersonAbstract person2) {  
 System.*out*.println(getName() + " встретился с " + person1 + " и " + person2);  
 }  
  
 @Override  
 public void talk(String people) {  
 System.*out*.println(getName() + " завизжал своим отвратительным голосом ");  
 }  
  
 @Override  
 public int haveMoney(int profit) throws OutOfMoneyError {  
 money += profit;  
 if (money >= 0){  
 return money;  
 }  
 else {  
 throw new OutOfMoneyError(getName() + " разорился. Деньги закончились!");  
 }  
 }  
  
 @Override  
 public void goTo(Locatoin locatoin) {  
 if (locatoin == Locatoin.*HOTEL*){  
 System.*out*.println(getName() + " появился в " + Locatoin.*HOTEL*.place);  
 }  
 }  
  
 public static class Clother {  
 static boolean *stick* = true;  
 static boolean *jacket* = true;  
 static boolean *pants* = true;  
 static boolean *hat* = true;  
 public static void wear(PersonAbstract person) {  
 System.*out*.println(person.getName() + " выглядит: черный пиджак- " + *jacket* + ", черный брюки- " + *pants* + ", черная высокая шляпа- " + *hat* + ", палка в руках- " + *stick*);  
 }  
 }  
 @Override  
 public boolean equals(Object o) {  
 if (o instanceof Superfild){  
 if (this.hashCode() == o.hashCode()) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

**Вывод программы:**

Имя: Спрутс. Бизнесс: самый успешный

Имя: Мигга. Бизнесс: неуспешный

Имя: Жулио. Бизнесс: неуспешный

Спрутс отправил телеграмму Крабс ->

Крабс получил телеграмму от Спрутс

Спрутс дал Мигга и Жулио 1000000

Бизнесс Жулио стал более успешный

Бизнесс Мигга стал более успешный

Бизнесс Спрутс стал менее успешный

Крабс знал: Спрутс действует разумно и осмотрительно

Крабс надеется: Спрутс даст Мигга и Жулио 2000000

Крабс ничего не сказал Мигге и Жулио

Крабс встретился с Мигга и Жулио

Жулио хочет удрать со всеми деньгами

Крабс решил не отправлять телеграмма

Крабс пошёл в ресторан:

Крабс пообедал

Крабс пошёл в гостиница:

Крабс поспал

Крабс искупался в бассейне

Крабс встретился с Мигга и Жулио

Крабс поужинал с друзьями

Мигга пошёл в театр:

Жулио пошёл в театр:

Крабс пошёл в театр:

В театре можно швырять в актёров гнилые яблоки.

Можно веселиться.

Крабс проснулся

Крабс ничего не сказал никому

Крабс отправил телеграмму Спрутс ->

Спрутс получил телеграмму от Крабс

Спрутс отправил телеграмму Крабс ->

Крабс получил телеграмму от Спрутс

ожидание 1 день

Крабс ничего не сказал Мигга и Жулио

Крабс отправил телеграмму Спрутс ->

Спрутс получил телеграмму от Крабс

Суперфильд появился в гостиница

Суперфильд выглядит: черный пиджак- true, черный брюки- true, черная высокая шляпа- true, палка в руках- true

Суперфильд столкнулся с Крабс

Суперфильд обнял Крабс

Суперфильд завизжал своим отвратительным голосом

**Вывод:**

Проделав данную лабораторную работу, я закрепила основные принципы ООП, принципы SOLID и изучила вложенные, локальные и анонимные классы, а также познакомилась с исключениями и их типами.