НИУ ИТМО

Факультет программной инженерии и компьютерной техники

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант № 121010.5

*Выполнил:*

Студент группы P3110

Абрабоу Ахмед Елсаид А.И

*Преподаватель:*Сорокин Роман Борисович



Санкт-Петербург, 2021

***Задание:***

***Graphical user interface, text, application, email

Description automatically generated***

***Explaining program***

Лабораторная работа 3, описание

Собственно говоря, в этой лабораторной работе представлена объектная модель, построенная на основе данного текста.

Текст: Нечего и говорить, что полицейские боялись теперь и близко подходить к ракете, а не то что стрелять возле нее. Деревенские жители могли беспрепятственно приходить к космонавтам и получать у них семена гигантских растений. Теперь гигантские семена сажали не только в деревне Нееловке, но и в селе Голопяткине, Бесхлебове, Голодаевке, Непролазном и во многих других. Знайка распорядился, чтоб лунатикам давали не только нужные им семена, но снабжали их приборами невесомости, а также антилунитом и объясняли им, как всем этим пользоваться, чтоб защититься от полицейских. Вскоре к космонавтам прибыли несколько рабочих со скуперфильдовской макаронной фабрики. Они сказали, что решили прогнать с фабрики Скуперфильда, а макароны будут делать сами без всяких хозяев. Чтоб осуществить этот план, им нужно устроить на фабрике невесомость, так как в противном случае полицейские могут помешать им и даже вовсе прогонят их с фабрики.

В классе Main содержится основная логика работы программы, в которой создаются нужные нам классы и вызываются методы из них. Сначала создаем класс Police, которые представляют «полицейских» в тексте, Villagers – «деревенские жители», Astronaut – «космонавты», Znayka – персонаж по имени «Знайка», FactoryWorkers – «рабочие с фабрики Скуперфильда», Scooperfield – сам «Скуперфильд», и, наконец, Factory – «фабрика Скуперфильда». Полицейские умеют бояться ракеты, подходить к ракете и стрелять возле неё (имеют соответствующие методы, далее все действия подразумевают под собой методы).

Деревенские жители имеют класс Seeds (семена, наследуется от Resource), WeightlessnessDevices (прибор невесомости, наследуется от Resource), Antilunite (антилуниты, наследуется от Resource). Они способны пойти к космонавтам (goToAstronauts с определенным именем), получить от них приборы невесомости (getWeightlessnessDevicesFromAstronaut с объектом прибора), антилунит (getAntiluniteFromAstronaut с объектом антилунита) и семена (так же), получить инструкции (getExplainFromAstronaut), рассказать про места посадки семян (Enum Village с названиями сел, tellAboutPlantingSites). Жители реализуют интерфейс GetHelpFromAstronauts, который позволяет им получить помощь от космонавтов.

Сами космонавты, соответственно, имеют в своем наличии все перечисленные выше ресурсы (поля класса), могут отдать все, что перечислено выше (три метода с первым словом give), объяснить, как этим пользоваться (explainHowToUse). (реализует интерфейс AstronautHelp, в которых содержатся их действия, можем на любую сущность нацепить)

Знайка может отдавать приказ (giveOrder()).

Рабочие могут прийти к космонавтам (arriveAtTheAstronauts), рассказать о своем плане (sayPlan), о их решении (tellWhatTheyDecided), подумать о решении (thinkAboutTheyDecided).

Скуперфильд является директором макаронной фабрики.

Фабрика может о себе рассказать, и получать/терять состояние невесомости (через поле weightlessness)

Вызываем все нужные нам методы по тексту, получаем исходный текст.

Лабораторная работа 4, описание

К лабе появились новые требования – необходимо придумать, как расположить два класса исключений, а так же добавить локальный, анонимный и вложенный класс.

Поэтому добавили unchecked класс-исключение NoGunException, которое у нас выбрасывается в случае, если по каким-то непонятным причинам у полицейских не оказалось оружия – тогда эта история не случится, потому что рабочим все равно на полицейских (как вариант развития событий). Unchecked потому, что у полицейских всегда есть оружие.

Также добавили checked класс-исключение NoResourcesException, означающее, что у нас закончились какие-либо ресурсы (вполне реальная ситуация, поэтому checked). В таком случае будем прерывать историю и говорить, что у космонавтов закончились ресурсы.

Далее, класс Scooperfield, представляющий одноименного персонажа, стал интерфейсом, для того чтобы мы могли в классе Main реализовать анонимный класс, который будет его представлять.

У полиции появился вложенный нестатический класс Gun, который является оружием для полицейских. Оружие умеет стрелять)

У Знайки в методе giveOrder() появился класс Order – приказ, который может отдать Знайка.

А для космонавтов был создан статический класс Instruction, который представляет инструкцию.

А так больше нет изменений

***Диограмма:***

Graphical user interface

Description automatically generated

***GitHub//:***

[***https://github.com/a0-0/ITMO-Programming-Labs/tree/master/prog%20lab4***](https://github.com/a0-0/ITMO-Programming-Labs/tree/master/prog%20lab4)

***Код программы:***

***Main class***

import buildings.Factory;  
import characters.\*;  
import exceptions.NoGunException;  
import exceptions.NoResourcesException;  
  
import java.util.Objects;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 Police police = new Police();  
 if (police.getGun() == null) throw new NoGunException(" у полиции нет оружия");  
 Villagers villagers = new Villagers();  
 Astronaut astronaut = new Astronaut();  
 Znayka znayka = new Znayka();  
 FactoryWorkers factoryWorkers = new FactoryWorkers();  
 Scooperfield scooperfield = new Scooperfield() {  
 private String placeOfWork = "Директор макаронной фабрики";  
 private String name = "Скуперфильд";  
  
 @Override  
 public void banishByWorkers() {  
 placeOfWork = "безработный";  
 }  
  
 @Override  
 public String toString() {  
 return "Scooperfield{" +  
 "placeOfWork='" + placeOfWork + '\'' +  
 ", name='" + name + '\'' +  
 '}';  
 }  
 };  
 Factory factory = new Factory();  
 System.*out*.print("Нечего и говорить, что ");  
 police.fearOfRocket();  
 villagers.goToAstronauts(astronaut.getName());  
 if (astronaut.getSeeds() == null) {  
 throw new NoResourcesException("нет семян");  
 }  
 villagers.getSeedsFromAstronaut(astronaut.giveSeedsToVillagers());  
 villagers.tellAboutPlantingSites();  
 znayka.giveOrder();  
 if (astronaut.getSeeds() == null || astronaut.getAntilunite() == null || astronaut.getWeightlessnessDevices() == null) {  
 throw new NoResourcesException("нет ресурсов");  
 }  
 villagers.getWeightlessnessDevicesFromAstronaut(astronaut.giveWeightlessnessDevicesToVillagers());  
 villagers.getAntiluniteFromAstronaut(astronaut.giveAntiluniteToVillagers());  
 villagers.getExplainFromAstronaut(new Astronaut.Instruction().getInstruction());  
 factoryWorkers.arriveAtTheAstronauts();  
 factoryWorkers.tellWhatTheyDecided();  
 factoryWorkers.thinkAboutTheyDecided();  
 } catch (NoGunException e) {  
 System.*out*.println("История не начнётся, т.к. у полиции нет пистолета, а значит рабочим он не страшны.");  
 } catch (NoResourcesException e) {  
 System.*out*.println("у космонавтов " + e.getMessage());  
 }  
 }  
}

***package buildings***

package buildings;  
  
import java.util.Objects;  
  
public class Factory {  
 private String nameOfFactory="макаронная фабрика";  
 private boolean weightlessness=false;  
  
 public boolean isWeightlessness() {  
 return weightlessness;  
 }  
  
 public void setWeightlessness(boolean weightlessness) {  
 this.weightlessness = weightlessness;  
 }  
  
 public String getNameOfFactory() {  
 return nameOfFactory;  
 }  
  
 public void setNameOfFactory(String nameOfFactory) {  
 this.nameOfFactory = nameOfFactory;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Factory factory = (Factory) o;  
 return weightlessness == factory.weightlessness && Objects.*equals*(nameOfFactory, factory.nameOfFactory);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(nameOfFactory, weightlessness);  
 }  
  
 @Override  
 public String toString() {  
 return "Factory{" +  
 "nameOfFactory='" + nameOfFactory + '\'' +  
 ", weightlessness=" + weightlessness +  
 '}';  
 }  
}

package buildings;  
public class Rocket {  
}

package buildings;  
  
public enum Village {  
 *GOLOPYATKINA*,  
 *BESKHLEBNOV*,  
 *GOLODAYEVKA*,  
 *IMPASSABLE*}

***Package characters***

package characters;  
  
import resources.Antilunite;  
import resources.Seeds;  
import resources.WeightlessnessDevices;  
  
import java.util.Objects;  
  
public class Astronaut implements AstronautHelp {  
 private final String name = "космонавтам";  
 private final String instruction= new Instruction().getInstruction();  
 private final Seeds seeds = new Seeds();  
 private final WeightlessnessDevices weightlessnessDevices = new WeightlessnessDevices();  
 private final Antilunite antilunite = new Antilunite();  
  
 public String getName() {  
 return name;  
 }  
 public static class Instruction{  
 private final String instruction = "Пользуйтесь так!";  
  
 public String getInstruction() {  
 return instruction;  
 }  
 }  
  
 public Seeds getSeeds() {  
 return seeds;  
 }  
  
 public WeightlessnessDevices getWeightlessnessDevices() {  
 return weightlessnessDevices;  
 }  
  
 public Antilunite getAntilunite() {  
 return antilunite;  
 }  
  
 @Override  
 public WeightlessnessDevices giveWeightlessnessDevicesToVillagers() {  
 return weightlessnessDevices;  
 }  
  
 @Override  
 public Seeds giveSeedsToVillagers() {  
 return seeds;  
 }  
  
 @Override  
 public Antilunite giveAntiluniteToVillagers() {  
 return antilunite;  
 }  
  
 public String explainHowToUse() {  
 return instruction;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Astronaut astronaut = (Astronaut) o;  
 return Objects.*equals*(name, astronaut.name) && Objects.*equals*(seeds, astronaut.seeds) && Objects.*equals*(weightlessnessDevices, astronaut.weightlessnessDevices) && Objects.*equals*(antilunite, astronaut.antilunite);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, seeds, weightlessnessDevices, antilunite);  
 }  
  
 @Override  
 public String toString() {  
 return "Astronaut{" +  
 "name='" + name + '\'' +  
 ", seeds=" + seeds +  
 ", weightlessnessDevices=" + weightlessnessDevices +  
 ", antilunite=" + antilunite +  
 '}';  
 }  
}

package characters;  
  
import resources.Antilunite;  
import resources.Seeds;  
import resources.WeightlessnessDevices;  
  
public interface AstronautHelp {  
 WeightlessnessDevices giveWeightlessnessDevicesToVillagers();  
  
 Seeds giveSeedsToVillagers();  
  
 Antilunite giveAntiluniteToVillagers();  
}

package characters;  
  
import java.util.Objects;  
  
public class FactoryWorkers extends Villagers {  
 private final String placeOfWork = "скуперфильдовской макаронной фабрики.";  
  
 public void arriveAtTheAstronauts() {  
 System.*out*.println("Вскоре к космонавтам прибыли несколько рабочих со " + placeOfWork);  
 }  
  
 public String sayPlan() {  
 return "прогнать с фабрики Скуперфильда, а макароны будут делать сами без всяких хозяев";  
 }  
  
 public void tellWhatTheyDecided() {  
 System.*out*.println("Они сказали, что решили " + sayPlan() + ".");  
 }  
  
 public void thinkAboutTheyDecided() {  
 System.*out*.println("Чтоб осуществить этот план, им нужно устроить на фабрике невесомость, " +  
 "так как в противном случае полицейские могут помешать им и даже вовсе прогонят их с фабрики.");  
 }  
  
 public boolean arrangeWeightlessness() {  
 return true;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 FactoryWorkers that = (FactoryWorkers) o;  
 return Objects.*equals*(placeOfWork, that.placeOfWork);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(placeOfWork);  
 }  
  
 @Override  
 public String toString() {  
 return "FactoryWorkers{" +  
 "placeOfWork='" + placeOfWork + '\'' +  
 '}';  
 }  
}

package characters;  
  
import resources.Antilunite;  
import resources.Seeds;  
import resources.WeightlessnessDevices;  
  
public interface GetHelpFromAstronaut {  
 void getWeightlessnessDevicesFromAstronaut(WeightlessnessDevices weightlessnessDevices);  
  
 void getAntiluniteFromAstronaut(Antilunite antilunite);  
  
 void getSeedsFromAstronaut(Seeds seeds);  
}

package characters;  
  
import java.util.Objects;  
  
public class Police {  
 private Gun gun = new Gun();  
  
 public Gun getGun() {  
 return gun;  
 }  
  
  
 private final String policeName = "полицейские";  
  
 public void fearOfRocket() {  
 System.*out*.println(policeName + " боялись теперь и близко " + goToRocket() + ", а не то что " + shootNearRocket() + ".");  
 }  
  
 public String goToRocket() {  
 return "подходить к ракете";  
 }  
  
 public String shootNearRocket() {  
 return gun.createShoot() + " возле неё";  
 }  
  
 public void preventEvent() {  
 }  
  
 public void banishFromFactory() {  
 }  
  
 public class Gun {  
 public String createShoot() {  
 return "стрелять";  
 }  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Police police = (Police) o;  
 return Objects.*equals*(policeName, police.policeName);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(policeName);  
 }  
  
 @Override  
 public String toString() {  
 return "Police{" +  
 "policeName='" + "полиция" + '\'' +  
 '}';  
 }  
}

package characters;  
  
public interface Scooperfield {  
 public void banishByWorkers();  
}

package characters;  
  
import buildings.Village;  
import resources.Antilunite;  
import resources.Seeds;  
import resources.WeightlessnessDevices;  
  
import java.util.Objects;  
  
public class Villagers implements GetHelpFromAstronaut {  
 private final String nameOfVillagers = "Деревенские жители";  
 private final String name = "лунтакам";  
 private Seeds seeds;  
 String instruction;  
 WeightlessnessDevices weightlessnessDevices;  
 Antilunite antilunite;  
  
 public void goToAstronauts(String name) {  
 System.*out*.print(nameOfVillagers + " могли беспрепятственно приходить к " + name + " ");  
 }  
  
 public void setWeightlessnessDevices(WeightlessnessDevices weightlessnessDevices) {  
 this.weightlessnessDevices = weightlessnessDevices;  
 }  
  
 public void setAntilunite(Antilunite antilunite) {  
 this.antilunite = antilunite;  
 }  
  
 public void setSeeds(Seeds seeds) {  
 this.seeds = seeds;  
 }  
  
 @Override  
 public void getWeightlessnessDevicesFromAstronaut(WeightlessnessDevices weightlessnessDevices) {  
 setWeightlessnessDevices(weightlessnessDevices);  
 }  
  
 @Override  
 public void getAntiluniteFromAstronaut(Antilunite antilunite) {  
 setAntilunite(antilunite);  
 }  
  
 @Override  
 public void getSeedsFromAstronaut(Seeds seeds) {  
 setSeeds(seeds);  
 System.*out*.println("и получать у них " + seeds.getName() + " гигантских растений.");  
 }  
  
 public void setInstruction(String instruction) {  
 this.instruction = instruction;  
 }  
  
 public void getExplainFromAstronaut(String instruction) {  
 setInstruction(instruction);  
 }  
  
 public void tellAboutPlantingSites() {  
 System.*out*.println("Теперь гигантские семена сажали не только в деревне Нееловке, но и в селе "  
 + Village.*GOLOPYATKINA*.name() + ", "  
 + Village.*BESKHLEBNOV*.name() + ", "  
 + Village.*GOLODAYEVKA*.name() + ", "  
 + Village.*IMPASSABLE*.name() + "и во многих других.");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Villagers villagers = (Villagers) o;  
 return Objects.*equals*(nameOfVillagers, villagers.nameOfVillagers) && Objects.*equals*(name, villagers.name) && Objects.*equals*(seeds, villagers.seeds) && Objects.*equals*(instruction, villagers.instruction) && Objects.*equals*(weightlessnessDevices, villagers.weightlessnessDevices) && Objects.*equals*(antilunite, villagers.antilunite);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(nameOfVillagers, name, seeds, instruction, weightlessnessDevices, antilunite);  
 }  
  
 @Override  
 public String toString() {  
 return "Villagers{" +  
 "nameOfVillagers='" + nameOfVillagers + '\'' +  
 ", name='" + name + '\'' +  
 ", seeds=" + seeds +  
 ", instruction='" + instruction + '\'' +  
 ", weightlessnessDevices=" + weightlessnessDevices +  
 ", antilunite=" + antilunite +  
 '}';  
 }  
}

package characters;  
  
public class Znayka {  
  
 public void giveOrder() {  
 class Order {  
 private String name = "Знайка";  
 private String order = "чтоб лунатикам давали не только нужные им семена, но снабжали их приборами невесомости," +  
 " а также антилунитом и объясняли им, как всем этим пользоваться, чтоб защититься от полицейских.";  
  
 public String createOrder() {  
 return name + " распорядился, " + order;  
 }  
 }  
 System.*out*.println(new Order().createOrder());  
 }  
  
}

***Package expiation***

package exceptions;  
  
public class NoGunException extends Exception{  
 public NoGunException(String message) {  
 super(message);  
 }  
}

package exceptions;  
  
public class NoResourcesException extends Exception{  
 public NoResourcesException(String message) {  
 super(message);  
 }  
}

***Package recourse***

package resources;  
  
import java.util.Objects;  
  
public class Antilunite extends Resource {  
 private String name="антилунит";  
 public Antilunite() {  
 super("антилунитом");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Antilunite that = (Antilunite) o;  
 return Objects.*equals*(name, that.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
  
 @Override  
 public String toString() {  
 return "Antilunite{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
}

package resources;  
  
public abstract class Resource {  
 private final String name;  
  
 public Resource(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
}

package resources;  
  
import java.util.Objects;  
  
public class Seeds extends Resource {  
 private String name="семена";  
 public Seeds() {  
 super("семена");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Seeds seeds = (Seeds) o;  
 return Objects.*equals*(name, seeds.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
  
 @Override  
 public String toString() {  
 return "Seeds{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
}

package resources;  
  
import java.util.Objects;  
  
public class Seeds extends Resource {  
 private String name="семена";  
 public Seeds() {  
 super("семена");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Seeds seeds = (Seeds) o;  
 return Objects.*equals*(name, seeds.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name);  
 }  
  
 @Override  
 public String toString() {  
 return "Seeds{" +  
 "name='" + name + '\'' +  
 '}';  
 }  
}

**Output:**

Нечего и говорить, что полицейские боялись теперь и близко подходить к ракете, а не то что стрелять возле неё.

Деревенские жители могли беспрепятственно приходить к космонавтам и получать у них семена гигантских растений.

Теперь гигантские семена сажали не только в деревне Нееловке, но и в селе GOLOPYATKINA, BESKHLEBNOV, GOLODAYEVKA, IMPASSABLEи во многих других.

Знайка распорядился, чтоб лунатикам давали не только нужные им семена, но снабжали их приборами невесомости, а также антилунитом и объясняли им, как всем этим пользоваться, чтоб защититься от полицейских.

Вскоре к космонавтам прибыли несколько рабочих со скуперфильдовской макаронной фабрики.

Они сказали, что решили прогнать с фабрики Скуперфильда, а макароны будут делать сами без всяких хозяев.

Чтоб осуществить этот план, им нужно устроить на фабрике невесомость, так как в противном случае полицейские могут помешать им и даже вовсе прогонят их с фабрики.

**Вывод:**

Во время выполнения лабораторной работы я укрепил свои знания в ООП, получил опыт в использовании внутренних статических, анонимных и локальных классов, а также в создании собственных исключений.