**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет**

**імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЗВІТ**

**З ЛАБОРАТОРНОЇ РОБОТИ №3**

з дисципліни«ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему **« Основи ООП моделювання на Java»**

**Варіант 15**

121 – ЛР.ПЗ.03 – 208.1810915

Виконала: студент 2 курсу, групи 209

спеціальності

121 «Інженерія програмного забезпечення»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.П. Кисса

(підпис,ініціали та прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Перевірив: викладач кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю. О. Нездолiй

(підпис,ініціали та прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

**м. Миколаїв – 2020 рік**

Завдання:

Змоделювати Космічну станцію = Піллігрім =, яка може приймати космічні кораблі використовуючи певні правила. Перше правило-на станції не може одночасно перебувати більше 10 кораблів. Кораблі можуть відповідно прилітати і відлітати зі станції. Кожен корабель управляється екіпажами з різних зоряних систем, різними рассами (турианца, земляни, ходакейци) .Туріанци не люблять ходакейцев, земляни не люблять турианца, ходакекйцам все одно. Відповідно це треба враховувати при прийомі кораблів. І останнє правило-станція має обмеження по сумарній масі кораблів, які нах. на станції, тобто якщо прибув корабель і з ним буде перевищено цей показник - його не можна прийняти. Треба формувати звіт по кораблям за період, які нах. на станції та  по кораблям, яким було відмовлено в доступі із зазначенням причини..

Class Rocket

package com.company;  
  
public class Rocket {  
 //Мощность ракеты  
 private int power;  
 //Масса коробля  
 private int massTon;  
 //Модель ракеты  
 private String model;  
  
 public Rocket(int power, String model, int massTon){  
 this.power=power;  
 this.model=model;  
 this.massTon=massTon;  
 }  
  
 public int getMass() { return massTon; }  
  
 public void setMass(int massTon) { this.massTon = massTon; }  
  
 public int getPower() {  
 return power;  
 }  
  
 public void setPower(int power) {  
 this.power = power;  
 }  
  
 public String getModel() {  
 return model;  
 }  
  
 public void setModel(String model) {  
 this.model = model;  
 }  
  
  
}

Class Humans

package com.company;  
  
public class Humans {  
 //Земляни управляют конкретной ракетой  
 private Rocket rocket;  
  
 public Humans(){}  
 public Humans(Rocket rocket){  
 this.rocket=rocket;  
 }  
 //Можно узнать какой ракетой он управляет  
 public Rocket getRocket() {  
 return rocket;  
 }  
 //Можно поручить управлять ракетой  
 public void setRocket(Rocket rocket) {  
 this.rocket = rocket;  
 }  
}

Class Turians

package com.company;

public class Turians {

//Турианци управляют конкретной ракетой

private Rocket rocket;

public Turians(){}

public Turians(Rocket rocket){

this.rocket=rocket;

}

//Можно узнать какой ракетой он управляет

public Rocket getRocket() {

return rocket;

}

//Можно поручить управлять ракетой

public void setRocket(Rocket rocket) {

this.rocket = rocket;

}

}

Class Hodakeys

package com.company;

public class Hodakeys {

//Ходакейци управляют конкретной ракетой

private Rocket rocket;

public Hodakeys(){}

public Hodakeys(Rocket rocket){

this.rocket=rocket;

}

//Можно узнать какой ракетой он управляет

public Rocket getRocket() {

return rocket;

}

//Можно поручить управлять ракетой

public void setRocket(Rocket rocket) {

this.rocket = rocket;

}

}

**КЛАСС БЫЛ ИЗМЕНЁН**

**А ИМЕННО ТЕПЕРЬ ТОЛЬКО ОДИН МЕТОД ПРИБЫТИЕ(arrive) и ТОЛЬКО ОДИН МЕТОД ОТБЫТИЯ(leaveTheStation)**

**Теперь используется приведение типов данных такое как**

**(Hodakeys) object, (Humans) object, (Turians) object, так как если не приводить тип object к выше перечисленым он просто будет ссылаться в никуда произойдёт NullPointerException.**

**И теперь только одна проверка на то сколько всего кораблей на станции** if (*getAmount*() > AMOUNT\_LIMIT) {  
 System.*out*.println("No more space for rockets sorry, we`ve got limit, limit is : " + getAmountLimit());  
 //Уменьшение счётчика  
 *amount*--;  
 } else {  
 System.*out*.println(*getAmount*());  
 }  
 }

Class Station

package com.company;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Station {  
 //Счёт ракет которые заходят на станцию  
 static int *amount* = 0;  
 //Счёт массы кораблей которые заходят на станцию  
 static int *currentWeight* = 0;  
  
 //Максимальный вес всех ракет которые находяться на станции  
 private final int TOTAL\_WEIGHT = 10000000;  
 //Максимальное количество ракет на станции  
 private final int AMOUNT\_LIMIT = 10;  
  
 //Рассы  
 private Turians turians = new Turians();  
 private Hodakeys hodakeys = new Hodakeys();  
 private Humans humans = new Humans();  
  
 //Коллекции  
 private List<Humans> humansList;  
 private List<Hodakeys> hodakeysList;  
 private List<Turians> turiansList;  
  
 public Station() {  
 humansList = new ArrayList<>();  
 hodakeysList = new ArrayList<>();  
 turiansList = new ArrayList<>();  
 }  
  
 public static int getAmount() {  
 return *amount*;  
 }  
  
 public static int getCurrentWeight() {  
 return *currentWeight*;  
 }  
  
 public int getTotalWeight() {  
 return TOTAL\_WEIGHT;  
 }  
  
 public int getAmountLimit() {  
 return AMOUNT\_LIMIT;

}

//Было удалено 3 метода, теперь 1 метод  
 //Прибытие на станцию ТЕПЕРЬ он принимает объект, который приводиться к тому типу данных которому захочет   
 public void arrive(Object object){

//Проверка если параметр который был передан в функцию и у него будет название класса Hoddakeys тогда будет выполняться данное условие.  
 if (object.getClass() == hodakeys.getClass()) {  
 //Если у корабля нет названия, или он неможет передвигаться, то есть у него нет мощности тогда их не пускает на станцию

Тут используется Приведение типов данных чтобы класс Object понял к какому объекту он должен ссылаться.   
 if (((Hodakeys) object).getRocket().getModel() == null || ((Hodakeys) object).getRocket().getPower() == 0) {  
 System.*out*.println("Model : " + ((Hodakeys) object).getRocket().getModel() + " Power : " + ((Hodakeys) object).getRocket().getPower());  
 }  
 //Есил масса корабля больше чем вся доступная масса станции тогда их не поскает на станцию  
 else if (((Hodakeys) object).getRocket().getMass() > TOTAL\_WEIGHT) {  
 System.*out*.println("Your Rocket is too heavy to our station");  
 }  
 //Если общая масса меньше чем масса всех кораблей + масса нового коробля, тогда их не пускает  
 else if (TOTAL\_WEIGHT < *currentWeight* + ((Hodakeys) object).getRocket().getMass()) {  
 System.*out*.println("Your rocket is too heavy, your rocket mass is " + ((Hodakeys) object).getRocket().getMass() + " Station total mass is: " + getTotalWeight() + " Our current mass is : " + *getCurrentWeight*());  
 }  
 //Если текущая масса меньше чем общая масса тогда их запускают на станцию  
 else {  
 if (*currentWeight* < TOTAL\_WEIGHT) {  
 //Добавление массы ракеты в текущую массу  
 *currentWeight* = ((Hodakeys) object).getRocket().getMass() + *currentWeight*;  
 System.*out*.println("Hodakeys comeIn" + *getCurrentWeight*());  
 hodakeysList.add((Hodakeys) object);  
 *amount*++;  
 } else {  
 System.*out*.println(*getCurrentWeight*() + " Current weight; " + getTotalWeight() + " Total weight");  
 }  
 }  
 }  
 else if (object.getClass() == humans.getClass()){  
 //Если у корабля нет названия, или он неможет передвигаться, то есть у него нет мощности тогда их не пускает на станцию  
 if (((Humans) object).getRocket().getModel() == null || ((Humans) object).getRocket().getPower() == 0) {  
 System.*out*.println("Model : " + ((Humans) object).getRocket().getModel() + " Power : " + ((Humans) object).getRocket().getPower());  
 }  
 //Если на станции есть ходакейци, тогда земляни не захотят останавливоться на этой станции  
 else if (hodakeysList.size() > 0) {  
 System.*out*.println("humans do not love hodakeys, ok?");  
 }  
 //Есил масса корабля больше чем вся доступная масса станции тогда их не поскает на станцию  
 else if (((Humans) object).getRocket().getMass() > TOTAL\_WEIGHT) {  
 System.*out*.println("Your Rocket is too heavy to our station");  
 }  
 //Если общая масса меньше чем масса всех кораблей + масса нового коробля, тогда их не пускает  
 else if (TOTAL\_WEIGHT < *currentWeight* + ((Humans) object).getRocket().getMass()) {  
 System.*out*.println("Your rocket is too heavy, your rocket mass is " + ((Humans) object).getRocket().getMass() + " Station total mass is: " + getTotalWeight() + " Our current mass is : " + *getCurrentWeight*());  
 }  
 //Если текущая масса меньше чем общая масса тогда их запускают на станцию  
 else {  
 if (*currentWeight* < TOTAL\_WEIGHT) {  
 //Добавление массы ракеты в текущую массу  
 *currentWeight* = ((Humans) object).getRocket().getMass() + *currentWeight*;  
 System.*out*.println("Humans comeIn" + *getCurrentWeight*());  
 humansList.add((Humans) object);  
 *amount*++;  
 } else {  
 System.*out*.println(*getCurrentWeight*() + " Current weight; " + getTotalWeight() + " Total weight");  
 }  
 }  
 }  
 else if (object.getClass() == turians.getClass()){  
 //Если у корабля нет названия, или он неможет передвигаться, то есть у него нет мощности тогда их не пускает на станцию  
 if (((Turians) object).getRocket().getModel() == null || ((Turians) object).getRocket().getPower() == 0) {  
 System.*out*.println("Model : " + ((Turians) object).getRocket().getModel() + " Power : " + ((Turians) object).getRocket().getPower());  
 }  
 //Если на станции есть земляни, тогда турианци не захотят останавливоться на этой станции  
 else if (humansList.size() > 0) {  
 System.*out*.println("Turians do not love humans, ok?");  
 }  
 //Если общая масса меньше чем масса всех кораблей + масса нового коробля, тогда их не пускает  
 else if (TOTAL\_WEIGHT < *currentWeight* + ((Turians) object).getRocket().getMass()) {  
 System.*out*.println("Your rocket is too heavy, your rocket mass is " + ((Turians) object).getRocket().getMass() + " Station total mass is: " + getTotalWeight() + " Our current mass is : " + *getCurrentWeight*());  
 }  
 //Если текущая масса меньше чем общая масса тогда их запускают на станцию  
 else {  
 if (*currentWeight* < TOTAL\_WEIGHT) {  
 //Добавление массы ракеты в текущую массу  
 *currentWeight* = ((Turians) object).getRocket().getMass() + *currentWeight*;  
 System.*out*.println("Turians comeIn" + *getCurrentWeight*());  
 turiansList.add((Turians) object);  
 *amount*++;  
 } else {  
 System.*out*.println(*getCurrentWeight*() + " Current weight; " + getTotalWeight() + " Total weight");  
 }  
 }  
 }

//ТОЛЬКО ОДНА ПРОВЕРКА НА ОГРАНИЧЕНИЕ ПО ВМЕСТИМОСТИ КОРАБЛЕЙ  
 //Максимум на станции может находиться 10 кораблей  
 if (*getAmount*() > AMOUNT\_LIMIT) {  
 System.*out*.println("No more space for rockets sorry, we`ve got limit, limit is : " + getAmountLimit());  
 //Уменьшение счётчика  
 *amount*--;  
 } else {  
 System.*out*.println(*getAmount*());  
 }  
 }  
//ТОЛЬКО ОДИН МЕТОД КОТОРЫЙ УДАЛЯЕТ ОБЪЕКТ ВМЕСТО ТРЁХ  
 //Покидают станцию  
 public void leaveTheStaton(Object object) {  
 if (object.getClass() == turians.getClass()) {  
 //Удаление элемента из листа  
 turiansList.remove(object);  
 //Уменьшение счётчика  
 *amount*--;  
 System.*out*.println("Bye tourians");  
 } else if (object.getClass() == hodakeys.getClass()) {  
 hodakeysList.remove(object);  
 *amount*--;  
 System.*out*.println("bye hodakeys");  
 } else if (object.getClass() == humans.getClass()) {  
 humansList.remove(object);  
 *amount*--;  
 System.*out*.println("byt humans");  
 }  
 }  
}

**КЛАСС БЫЛ ИЗМЕНЁН**

**Вместо humanArrives, turiansArrives, hodakeysArrive, только метод arrive**

**Вместо removeHumans, removeTurians, removeHodakyes, только метод leaveTheStation**

Class Main

package com.company;  
  
  
  
public class Main {  
  
 public static void main(String[] args) {  
 //Создание станции  
 Station station = new Station();  
  
 //Создание ракет, Агрегация  
 Rocket humanRocket = new Rocket(1600, "humanRocket", 1000000);  
 Rocket hodakeysRocket = new Rocket(2000, "hodakeysRocket", 1000000);  
 Rocket turiansRocket = new Rocket(1200, "turiansRocket", 1);  
  
 //Создание экипажа для ракет  
 Humans humans = new Humans(humanRocket);  
 Hodakeys hodakeys = new Hodakeys(hodakeysRocket);  
 Turians turians = new Turians(turiansRocket);  
  
 station.arrive(hodakeys);  
 station.arrive(humans);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
 station.arrive(turians);  
   
 station.arrive(turians);  
   
 station.leaveTheStaton(turians);  
 }  
}

Результат роботи програми

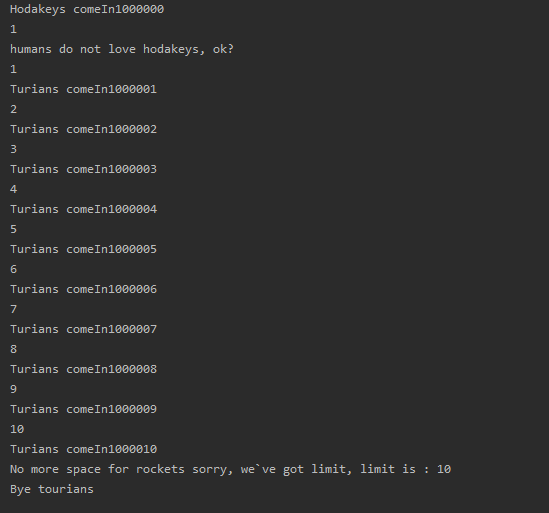


Рисунок 1. Результат роботи програми

Композиция — отношение, когда объект является частью другого объекта и не может принадлежат еще кому-то.

Создаем отношение в конструкторе.

Агрегация — отношение когда один объект является частью другого.

Создаем отношение где-то в коде и используем его как параметр для объкта.

Наследование механизм, который позволяет описать новый класс на основе существующего. При этом свойства и функциональность родительского класса заимствуются новым классом. Ключевое слово extends

Class A{

Private int a;

Public class (int a ){

This.a=a;

}

}

Class B extends A{

Public B(int a){

Super(a)

}

}

Инкапсуляция это сокрытие данных, переменные класа будут скрыты от других классов и доступ к ним может быть получен модификаторами доступа и геттерами и сеттерами