**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет**

**імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЗВІТ**

**З ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни«ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему **«  Створення графічного інтерфейсу до моделі з лабораторної №3»**

**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**

Class Main

package sample;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
  
 @Override  
 public void start(Stage primaryStage) throws Exception{  
 Parent root = FXMLLoader.*load*(getClass().getResource("sample.fxml"));  
 primaryStage.setTitle("Station javaFx");  
 primaryStage.setScene(new Scene(root, 800, 800));  
 primaryStage.show();  
 }  
  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

Class Controller

package sample;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.ResourceBundle;  
  
public class Controller{  
  
 //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 = 11;  
  
 //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;  
  
 @FXML  
 private Button addRocket;  
  
 @FXML  
 private TextField model;  
  
 @FXML  
 private TextField massTon;  
  
 @FXML  
 private TextField power;  
  
 @FXML  
 private TextField rocketModelToDelete;  
  
 @FXML  
 private Button deleteRocket;  
  
 @FXML  
 private ListView<Turians> turiansListView;  
  
 @FXML  
 private ListView<Humans> humanListView;  
  
 @FXML  
 private ListView<Hodakeys> hodakeysListView;  
  
 public Controller() {  
 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;  
 }  
  
  
  
  
  
 @FXML  
 void initialize() {  
 addRocket.setOnAction(actionEvent -> {  
 createRocket();  
 });  
  
 deleteRocket.setOnAction(actionEvent -> {  
 removeRocket();  
 });  
  
 }  
  
 public void removeRocket() {  
 String rocketModeltoDelete = model.getText();  
  
 if(rocketModeltoDelete.contains("Turians") || rocketModeltoDelete.contains("turians")){  
 turiansListView.getItems().remove(turians);  
 leaveTheStaton(turians);  
 }  
 else if (rocketModeltoDelete.contains("Humans") || rocketModeltoDelete.contains("humans")){  
 humanListView.getItems().remove(humans);  
 leaveTheStaton(humans);  
 }  
 else if (rocketModeltoDelete.contains("Hodakeys") || rocketModeltoDelete.contains("hodakeys")){  
 hodakeysListView.getItems().remove(hodakeys);  
 leaveTheStaton(hodakeys);  
 }  
 else {  
 System.*out*.println("There no model like that" + model.getText());  
 }  
 }  
 public void createRocket() {  
 String rocketModel = model.getText();  
 int rocketPower = Integer.*parseInt*(String.*valueOf*(power.getText()));  
 int rocketMass = Integer.*parseInt*(String.*valueOf*(massTon.getText()));  
 if (rocketMass > TOTAL\_WEIGHT) {  
 System.*out*.println("ur rocket is too heavy");  
 } else if (TOTAL\_WEIGHT < *currentWeight* + rocketMass) {  
 System.*out*.println("your rcoket is too heavy for us v2");  
 } //else if (getAmount() > AMOUNT\_LIMIT){  
// System.out.println("no more space for you bae");  
// }  
 else {  
 if (rocketModel.contains("Turians") || rocketModel.contains("turians")) {  
 if (humanListView.getItems().size() > 0){  
 System.*out*.println("turians hate humans");  
 }  
 else {  
 Rocket turinasRocket = new Rocket(rocketPower, rocketModel, rocketMass);  
 turians = new Turians(turinasRocket);  
 arrive(turians);  
 turiansListView.getItems().add(turians);  
 }  
 } else if (rocketModel.contains("Humans") || rocketModel.contains("humans")) {  
 if (hodakeysListView.getItems().size() > 0){  
 System.*out*.println("Humans hate hodakeys");  
 }  
 else {  
 Rocket humansRocket = new Rocket(rocketPower, rocketModel, rocketMass);  
 humans = new Humans(humansRocket);  
 arrive(humans);  
 humanListView.getItems().add(humans);  
 }  
 } else if (rocketModel.contains("Hodakeys") || rocketModel.contains("hodakeys")) {  
 Rocket hodakeysRocket = new Rocket(rocketPower, rocketModel, rocketMass);  
 hodakeys = new Hodakeys(hodakeysRocket);  
 arrive(hodakeys);  
 hodakeysListView.getItems().add(hodakeys);  
 } else {  
 System.*out*.println("wrong race");  
 }  
 }  
 if (*getAmount*() >= AMOUNT\_LIMIT) {  
 if (rocketModel.contains("Turians") || rocketModel.contains("turians")){  
 turiansListView.getItems().remove(turians);  
 *amount*--;  
 }  
 else if (rocketModel.contains("Humans") || rocketModel.contains("humans")){  
 humanListView.getItems().remove(humans);  
 *amount*--;  
 }  
 else if (rocketModel.contains("Hodakeys") || rocketModel.contains("hodakeys")){  
 hodakeysListView.getItems().remove(hodakeys);  
 *amount*--;  
 }  
 } else {  
 System.*out*.println(*getAmount*());  
 }  
 System.*out*.println("ROCKET is created for " + model.getText());  
 }  
  
 public void arrive(Object object){  
 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  
 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");  
 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");  
 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");  
 turiansList.add((Turians) object);  
 *amount*++;  
 } else {  
 System.*out*.println(*getCurrentWeight*() + " Current weight; " + getTotalWeight() + " Total weight");  
 }  
 }  
 }  
 //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*());  
 }  
 }  
  
 //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");  
 }  
 }  
  
}

Class Humans

package sample;  
  
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 sample;  
  
  
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;  
 }  
  
 @Override  
 public String toString() {  
 return "Turians{" +  
 "rocket=" + rocket +  
 '}';  
 }  
}

Class Hodakeys

package sample;  
  
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;  
 }  
}

Class Rocket

package sample;  
  
public class Rocket {  
 //Мощность ракеты  
 private int power = 1000;  
 //Масса коробля  
 private int massTon = 1000;  
 //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;  
 }  
  
 @Override  
 public String toString() {  
 return "Rocket{" +  
 "power=" + power +  
 ", massTon=" + massTon +  
 ", model='" + model + '\'' +  
 '}';  
 }  
}

sample.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ListView?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="800.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">  
 <children>  
 <Button fx:id="addRocket" layoutX="72.0" layoutY="178.0" mnemonicParsing="false" prefHeight="45.0" prefWidth="84.0" text="add rocket" />  
 <TextField fx:id="model" layoutX="39.0" layoutY="93.0" promptText="Model" />  
 <TextField fx:id="massTon" layoutX="39.0" layoutY="134.0" promptText="Mass" />  
 <TextField fx:id="power" layoutX="39.0" layoutY="51.0" promptText="Power" />  
 <TextField fx:id="rocketModelToDelete" layoutX="265.0" layoutY="46.0" prefHeight="35.0" prefWidth="164.0" promptText="input rocket model" />  
 <Button fx:id="deleteRocket" layoutX="286.0" layoutY="101.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="91.0" text="delete rocket" />  
 <ListView fx:id="turiansListView" layoutX="39.0" layoutY="264.0" prefHeight="237.0" prefWidth="234.0" />  
 <ListView fx:id="humanListView" layoutX="283.0" layoutY="264.0" prefHeight="237.0" prefWidth="234.0" />  
 <ListView fx:id="hodakeysListView" layoutX="531.0" layoutY="264.0" prefHeight="237.0" prefWidth="234.0" />  
 </children>  
</AnchorPane>