МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине СПП

Тема: «Классы в программах на языке программирования Java»

Выполнил

студент группы ПО-5

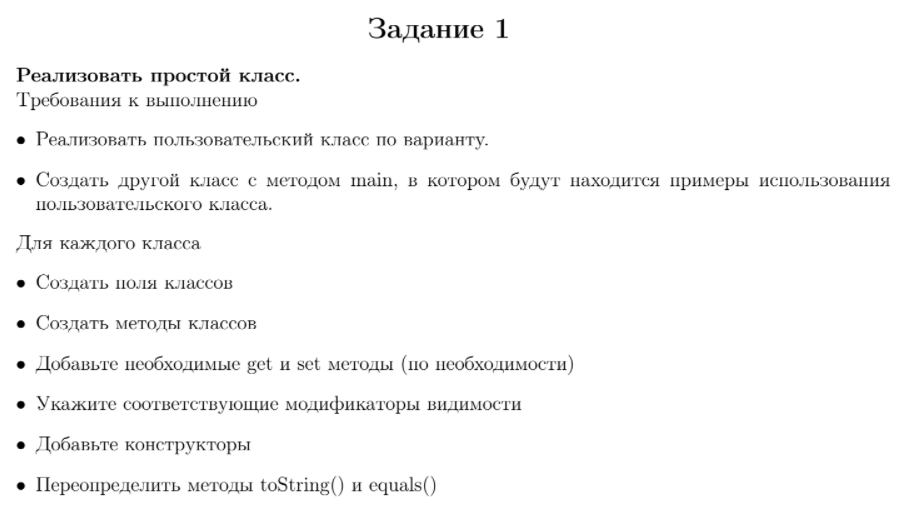
Романюк. В. А.

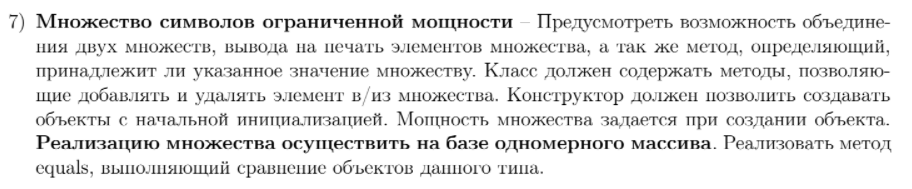
Проверил: преподаватель

Крощенко А. А.

Брест, 2021

Цель работы: научиться создавать и использовать классы в программах на языке программирования Java.

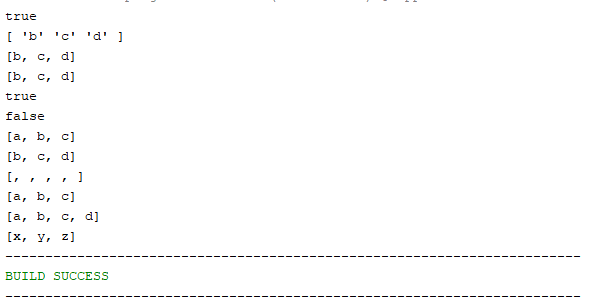


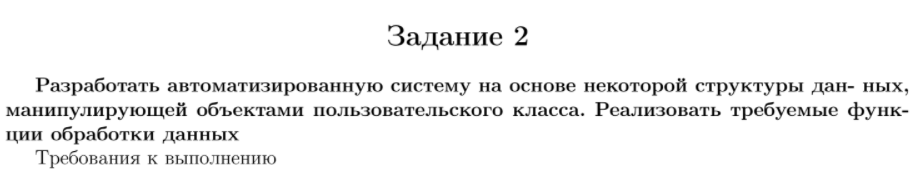


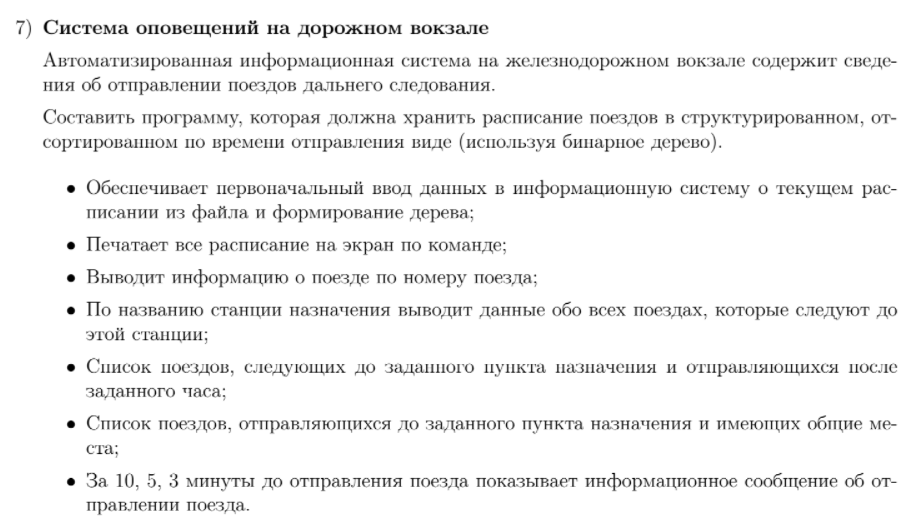
Код программы

package com.company;  
  
import java.util.HashSet;  
import java.util.Arrays;  
  
public class Lab3Task1 {  
 public static void main(String[] args) {  
 char[] mass = {'a', 'b', 'c'};  
 Symbols obj1 = new Symbols(mass, mass.length);  
  
 Symbols obj2 = new Symbols(mass, 3);  
  
 Symbols obj3 = new Symbols(5);  
  
 System.*out*.println(obj1.equals(obj2));  
  
 obj2.remove('a');  
 obj2.add('d');  
 obj1.add('b');  
  
 obj2.print();  
 System.*out*.println(obj2);  
 System.*out*.println(obj2.toString());  
  
 System.*out*.println(obj2.contains('c'));  
 System.*out*.println(obj2.contains('x'));  
  
 System.*out*.println(obj1);  
 System.*out*.println(obj2);  
 System.*out*.println(obj3);  
  
 obj2.append(obj1);  
  
 System.*out*.println(obj1);  
 System.*out*.println(obj2);  
  
 char[] mass2 = {'x', 'y', 'z'};  
  
 obj1.setCharSet(mass2);  
  
 System.*out*.println(obj1);  
  
  
 }  
}  
  
class Symbols {  
  
 private char[] charSet;  
 private int maxSize;  
  
 public Symbols(int length) {  
 this.maxSize = length;  
 this.charSet = new char[0];  
 }  
  
 public Symbols(char[] charSet, int length) {  
 HashSet<Character> charHashSet = convertHash(charSet);  
  
 if (charHashSet.size() <= length) {  
 this.maxSize = length;  
 this.charSet = new char[charHashSet.size()];  
  
 int count = 0;  
 for (char i : charHashSet) {  
 this.charSet[count] = i;  
 count++;  
 }  
 }  
 else {  
 System.*out*.println("Длина множества превышает указанный максимум");  
 this.maxSize = length;  
 this.charSet = new char[0];  
 }  
 }  
  
 public HashSet<Character> convertHash(char[] charSet) {  
 HashSet<Character> charHashSet = new HashSet<>();  
  
 for (int i = 0; i < charSet.length; i++) {  
 charHashSet.add(charSet[i]);  
 }  
  
 return charHashSet;  
 }  
  
 public char[] getCharSet() {  
 return this.charSet;  
 }  
  
 public void setCharSet(char[] charSet) {  
 HashSet<Character> charHashSet = convertHash(charSet);  
  
 if (charHashSet.size() <= this.maxSize) {  
 this.charSet = new char[charHashSet.size()];  
  
 int count = 0;  
 for (char i : charHashSet) {  
 this.charSet[count] = i;  
 count++;  
 }  
 }  
 else {  
 System.*out*.println("Недопустимая длина множества, превышен максимум");  
 }  
 }  
  
 @Override  
 public String toString() {  
 return Arrays.*toString*(this.charSet);  
 }  
  
 public boolean equals(Symbols obj) {  
 return Arrays.*equals*(this.charSet, obj.charSet);  
 }  
  
 public void append(Symbols obj) {  
 HashSet<Character> charHashSet = convertHash(charSet);  
  
 for (int i = 0; i < obj.charSet.length; i++) {  
 charHashSet.add(obj.charSet[i]);  
 }  
  
 if (charHashSet.size() <= this.maxSize) {  
 this.charSet = new char[charHashSet.size()];  
  
 int count = 0;  
 for (char i : charHashSet) {  
 this.charSet[count] = i;  
 count++;  
 }  
 }  
 else {  
 System.*out*.println("Множество, получаемое в результате объединения, имеет длину, превышающую максимальную");  
 }  
 }  
  
 public void print() {  
 System.*out*.print("[ ");  
  
 for (char i : this.charSet) {  
 System.*out*.print("'" + i + "' ");  
 }  
  
 System.*out*.println("]");  
 }  
  
 public boolean contains(char symbol) {  
 boolean hasSymbol = false;  
  
 for (char i : this.charSet) {  
 if (i == symbol) {  
 hasSymbol = true;  
 break;  
 }  
 }  
  
 return hasSymbol;  
 }  
  
 public void add(char symbol) {  
 HashSet<Character> charHashSet = convertHash(charSet);  
  
 charHashSet.add(symbol);  
  
 if (charHashSet.size() <= this.maxSize) {  
 this.charSet = new char[charHashSet.size()];  
  
 int count = 0;  
 for (char i : charHashSet) {  
 this.charSet[count] = i;  
 count++;  
 }  
 }  
 else {  
 System.*out*.println("Множество достигло максимальной длины");  
 }  
 }  
  
 public void remove(char symbol) {  
 HashSet<Character> charHashSet = new HashSet<>();  
  
 for (int i = 0; i < this.charSet.length; i++) {  
 charHashSet.add(this.charSet[i]);  
 }  
  
 charHashSet.remove(symbol);  
  
 this.charSet = new char[charHashSet.size()];  
  
 int count = 0;  
 for (char i : charHashSet) {  
 this.charSet[count] = i;  
 count++;  
 }  
 }  
}

Тестирование



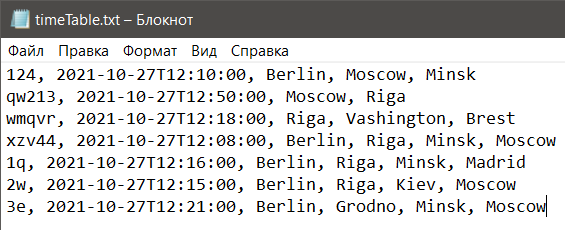




Код программы

package com.company;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.TreeMap;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.time.LocalDateTime;  
import static java.time.temporal.ChronoUnit.SECONDS;  
import java.util.HashSet;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class Lab3Ex2 extends TimerTask {  
 static private TreeMap<Train, LocalDateTime> *trains*;  
  
 @Override  
 public void run() {  
 *trains*.keySet().forEach(item -> {  
 if (((LocalDateTime.now().until(item.getTime(), SECONDS)) < 601 &&  
 (LocalDateTime.now().until(item.getTime(), SECONDS)) > 599) && !item.getPrintedLeave(0)) {  
 System.out.println("The train \"" + item.getName() + "\" leaves in 10 minutes");  
 item.setPrintedLeave(0, true);  
 }  
 else if (((LocalDateTime.now().until(item.getTime(), SECONDS)) < 301 &&  
 (LocalDateTime.now().until(item.getTime(), SECONDS)) > 299) && !item.getPrintedLeave(1)) {  
 System.out.println("The train \"" + item.getName() + "\" leaves in 5 minutes");  
 item.setPrintedLeave(1, true);  
 }  
 else if (((LocalDateTime.now().until(item.getTime(), SECONDS)) < 181 &&  
 (LocalDateTime.now().until(item.getTime(), SECONDS)) > 179) && !item.getPrintedLeave(2)) {  
 System.out.println("The train \"" + item.getName() + "\" leaves in 3 minutes");  
 item.setPrintedLeave(2, true);  
 }  
 });  
 }  
  
 public static void main(String[] args) {  
 try {  
 *trains* = new TreeMap<>(Comparator.comparing(o -> o.getTime()));  
  
 BufferedReader bfr = new BufferedReader(new FileReader(args[0]));  
 String str;  
 String[] subStrings;  
  
 while((str = bfr.readLine()) != null) {  
 subStrings = str.split(", ");  
  
 ArrayList<String> trainPoints = new ArrayList();  
 for (int i = 2; i < subStrings.length; i++) {  
 trainPoints.add(subStrings[i]);  
 }  
  
 String[] trainPointsMass = new String[trainPoints.size()];  
  
 Train train = new Train(subStrings[0], LocalDateTime.parse(subStrings[1]), trainPoints.toArray(trainPointsMass));  
 *trains*.put(train, train.getTime());  
 }  
  
 TimerTask timerTask = new Lab3Ex2();  
 Timer timer = new Timer(true);  
 timer.schedule(timerTask, 0, 1000);  
  
 *printTree*();  
  
 System.out.println();  
  
 *printSimilarTrainsByEndPoint*("Riga");  
  
 try {  
 Thread.sleep(1000 \* 10 \* 60);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 timer.cancel();  
 }  
 catch(IOException e) {  
 System.out.println("Ошибка!\n" + e.getMessage());  
 }  
 }  
  
 public static void printTree() {  
 *trains*.keySet().forEach(item -> {  
 item.print();  
 });  
 }  
  
 public static void printTrain(String name) {  
 *trains*.keySet().forEach(item -> {  
 if (item.getName().equals(name)) {  
 item.print();  
 }  
 });  
 }  
  
 public static void printTrainsByEndPoint(String point) {  
 *trains*.keySet().forEach(item -> {  
 boolean hasPoint = false;  
  
 for (String trainPoint : item.getPoints()) {  
 if (trainPoint.equals(point)) {  
 hasPoint = true;  
 break;  
 }  
 }  
  
 if (hasPoint) {  
 item.print();  
 }  
 });  
 }  
  
 public static void printTrainsByEndPointAndTime(String point, String time) {  
 *trains*.keySet().forEach(item -> {  
 if (item.getTime().isAfter(LocalDateTime.parse(time))) {  
 boolean hasPoint = false;  
  
 for (String trainPoint : item.getPoints()) {  
 if (trainPoint.equals(point)) {  
 hasPoint = true;  
 break;  
 }  
 }  
  
 if (hasPoint) {  
 item.print();  
 }  
 }  
 });  
 }  
  
 public static void printSimilarTrainsByEndPoint(String point) {  
 HashSet<Train> sameEndPointTrains = new HashSet<>();  
 HashSet<String> points = new HashSet<>();  
 HashSet<Train> similarTrains = new HashSet<>();  
  
 *trains*.keySet().forEach(item -> {  
 boolean hasPoint = false;  
  
 for (String elem : item.getPoints()) {  
 if (elem.equals(point)) {  
 hasPoint = true;  
 break;  
 }  
 }  
  
 if (hasPoint) {  
 sameEndPointTrains.add(item);  
 }  
 });  
  
 System.out.println("Endpoint: " + point);  
  
 sameEndPointTrains.forEach(item -> {  
 for (String trainPoint : item.getPoints()) {  
 similarTrains.add(item);  
  
 if (!points.contains(trainPoint) && !trainPoint.equals(point)) {  
 sameEndPointTrains.forEach(item2 -> {  
 if (!item2.getName().equals(item.getName())) {  
 for (String trainPoint2 : item2.getPoints()) {  
 if (trainPoint.equals(trainPoint2)) {  
 similarTrains.add(item2);  
 }  
 }  
 }  
 });  
 }  
  
 if (similarTrains.size() > 1) {  
 System.out.println("\tPoint: " + trainPoint);  
  
 similarTrains.forEach(elem -> {  
 System.out.print("\t");  
 elem.print();  
 });  
  
 System.out.println();  
 }  
  
 points.add(trainPoint);  
 similarTrains.clear();  
 }  
 });  
 }  
}  
  
class Train {  
 private String name;  
 private String[] points;  
 private LocalDateTime time;  
 private boolean[] printedLeave = {false, false, false};  
  
 public Train(String name, LocalDateTime time, String[] points) {  
 this.name = name;  
 this.time = time;  
 this.points = points;  
 }  
  
 public void print() {  
 String message = "The train \"" + this.name + "\" leaves at " + this.time.toString() + " in the direction [";  
  
 for (int i = 0; i < points.length; i++) {  
 message += points[i];  
 if (i != points.length - 1) {  
 message += ", ";  
 }  
 }  
  
 message += "]";  
  
 System.out.println(message);  
 }  
  
 public String getName() {  
 return this.name;  
 }  
  
 public String[] getPoints() {  
 return this.points;  
 }  
  
 public LocalDateTime getTime() {  
 return this.time;  
 }  
  
 public boolean getPrintedLeave(int index) {  
 return this.printedLeave[index];  
 }  
  
 public void setPrintedLeave(int index, boolean value) {  
 this.printedLeave[index] = value;  
 }  
}

Тестирование







Вывод: научился создавать и использовать классы в программах на языке программирования Java.