Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інженерно-хімічний факультетКафедра «Технічних та програмних засобів автоматизації»

Розрахункова робота

Програмування-2. Об'єктно-орієнтоване програмування

Варіант 21

Інформаційна система «Атракціон»

Виконав: студент гр. ЛА-03, Сенюк М.П.

Перевірив: к.т.н., доцент Д.О. Ковалюк

Київ 2021

ЗМІСТ

1. Технічне завдання.
2. Розділ 1 – Опис предметної області.
3. Розділ 2 – Структура програмного забезпечення.
4. Висновки.
5. Додатки.

**Технічне завдання**

Інформаційна система «Атракціон». Функції:

* сортування квитків за прізвищем
* сортування квитків за типом атракціону
* вивести всі квитки для певного типу атракціону
* додавання, редагування, видалення квитків
* завантаження та вивантаження даних у CSV форматі

Клас «квиток» має містити наступні поля:

* прізвище – строка
* тип атракціону – фіксований набір значень (тир, карусель, оглядове колесо)
* вартість квитка - ціле число
* дата придбання - дата

**Розділ 1**

Сутність «Атракціон» - атрибути:

* Прізвище покупця;
* Тип атракціону – (тир, карусель, оглядове колесо);
* Дата придбання;
* Вартість квитка;

Сутність «Атракціон» - методи:

* Отримання списку клієнтів;
* Отримання списку білетів;
* Додавання білетів з вводом дати, прізвища та типу атракціону;
* Сортування за категоріями та ім’ям;

**Розділ 2**

ClassTicket– містить інформацію про страховий поліс.

|  |  |
| --- | --- |
| private String secondName; | Прізвище білета; |

|  |  |
| --- | --- |
| private int cost; | Ціна білету; |
| AttractionType; | Тип атракціону – (тир, карусель, оглядове колесо); |
| GregorianCalendar ticketDate = new GregorianCalendar(); | Дата видачі; |
| public static Comparator<Ticket > sortingByName = new Comparator<Ticket >() | Сортування списку клієнтів за прізвищем; |
| public static Comparator< > sortingByCategory = new Comparator<Ticket >() | Сортування списку за категорією (тир, карусель, оглядове колесо); |

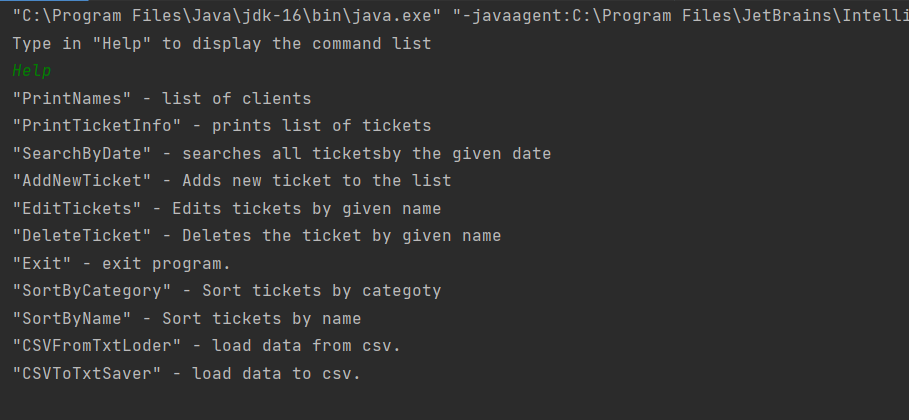
Ticket– менеджер роботи з клієнтами та полісами.

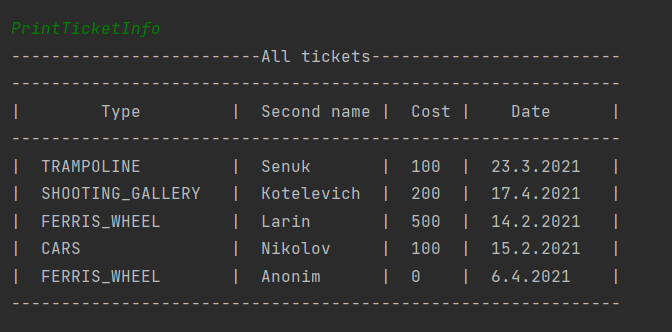
|  |  |
| --- | --- |
| public static void printListName(ArrayList<Ticket > list) | Виведення відсортованого за алфавітом списку клієнтів за прізвищем; |
| public static void printListCategory(ArrayList<Ticket> list) | Виведення відсортованого за категорією (а списку; |
| public static void editTicket (Ticket p) | Редагування страхового полісу; |
| ArrayList<Ticket > client = new ArrayList<Ticket>(); | Список квитків. |

**Висновок**

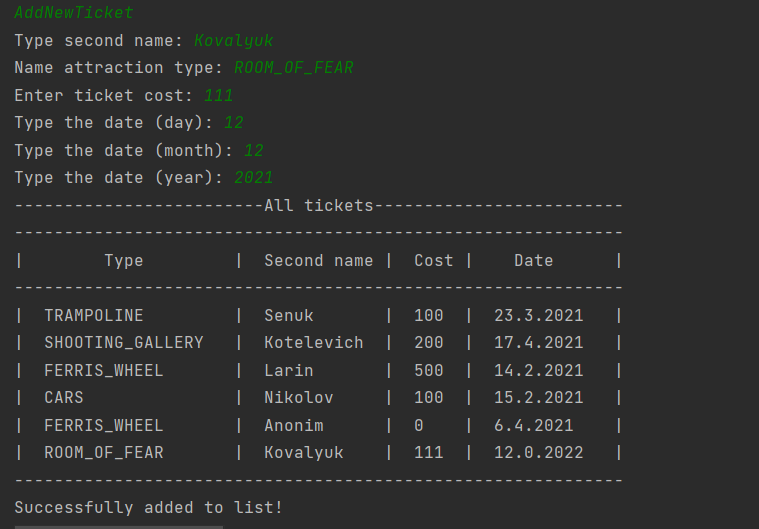
В даній роботі:

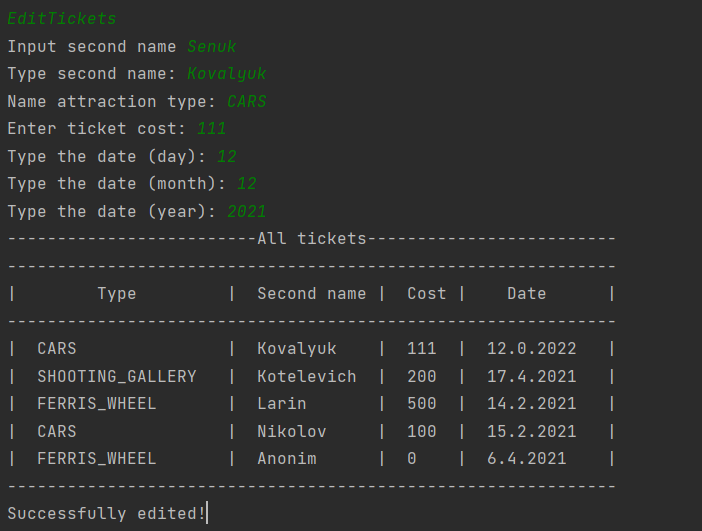
1. Реалізовано роботу в інтерактивному режимі з використанням меню;
2. Реалізовано роботу з колекціями, зокрема операції додавання, редагування, видалення, пошук, сортування, завантаження та вивантаження даних у CSV форматі.

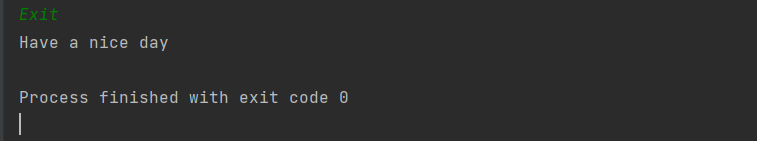
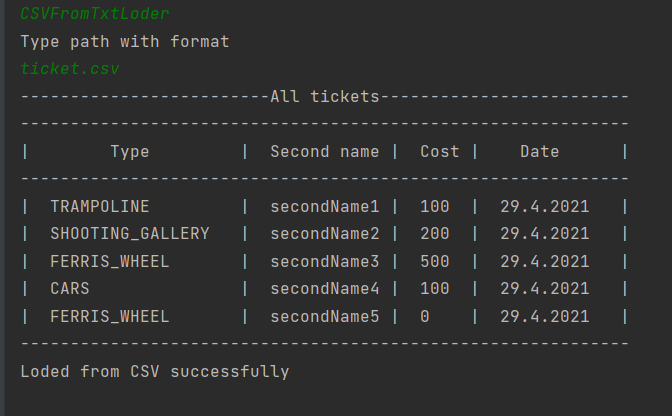
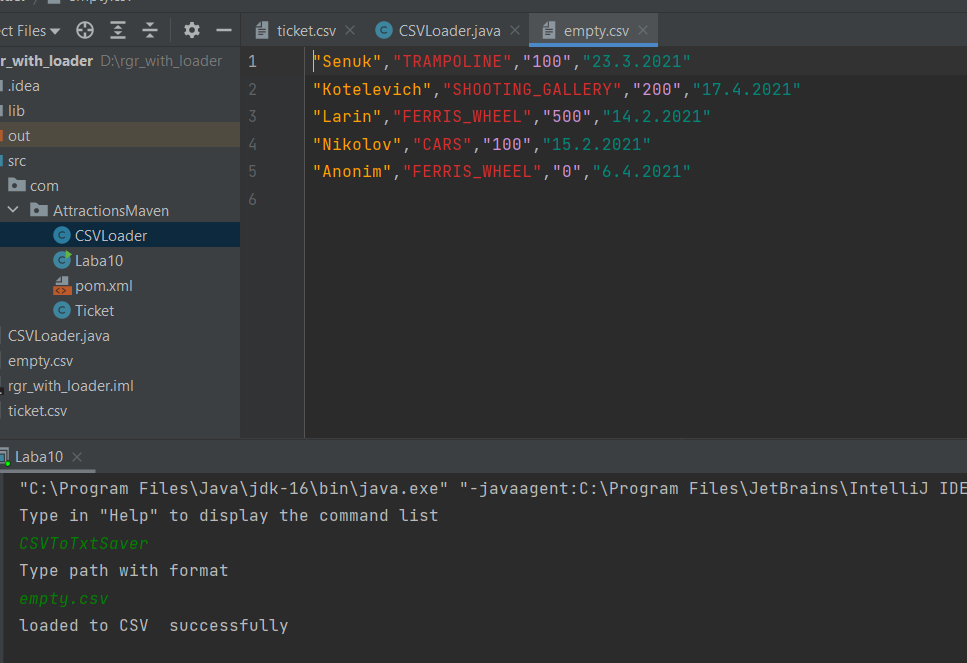
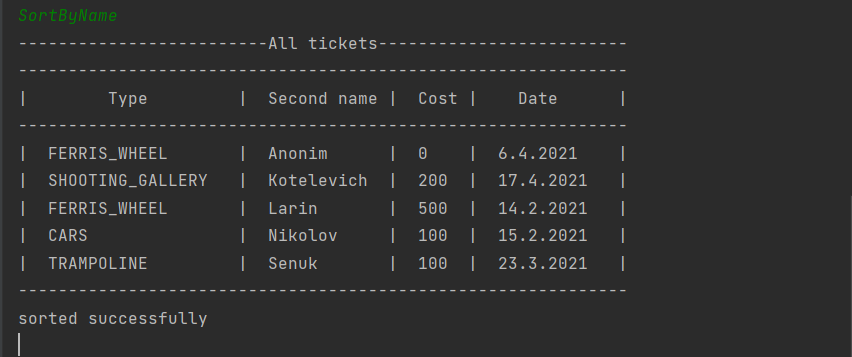
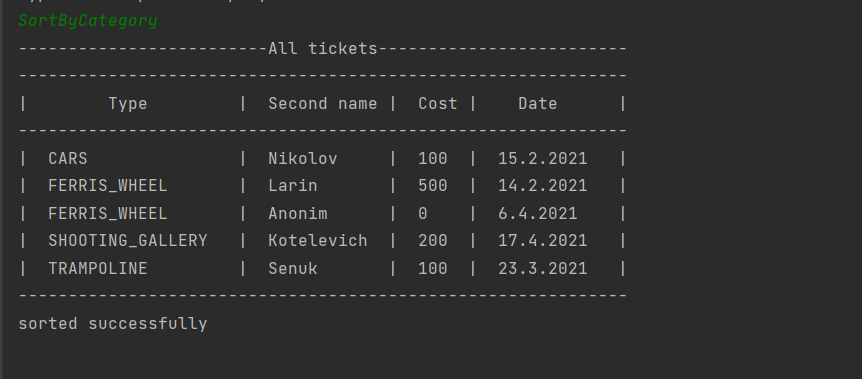
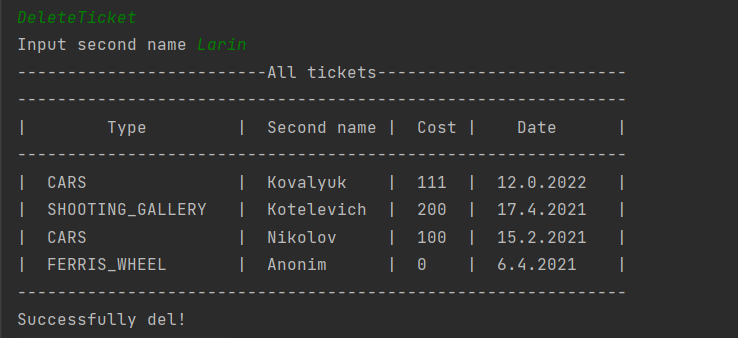












**Додаток – Лістинг програми**

package com.AttractionsMaven;  
  
import java.io.File;  
import java.util.ArrayList;  
import java.util.GregorianCalendar;  
import java.util.Scanner;  
  
public class Laba10 {  
 public static void printListName(ArrayList<Ticket> list) {  
 for(Ticket p : list) {  
 System.*out*.println(p.getSecondName());  
 }  
 }  
   
 public static void printListCategory(ArrayList<Ticket> list) {  
 System.*out*.println("-------------------------All tickets-------------------------");  
 System.*out*.println("-------------------------------------------------------------");  
 System.*out*.printf("|%-8s%-13s%-3s%-6s%-2s%-5s%-5s%-10s%s\n", "" ,"Type", "|" ,"Second name " ,"|" ,  
 " Cost ", "|" ,"Date", "|" );  
 System.*out*.println("-------------------------------------------------------------");  
 for(Ticket p : list) {  
  
 System.*out*.printf("| %-19s| %-12s| %-5d| %-12s|\n", p.getCategory(),  
 p.getSecondName(), p.getCost(), p.getDate());  
 }  
 System.*out*.println("-------------------------------------------------------------");  
  
  
 }  
   
 public static void editTicket(Ticket p) {  
 try {  
 Scanner commandString = new Scanner(System.*in*);  
 System.*out*.print("Type second name: ");  
 String secondName = commandString.nextLine();  
   
 System.*out*.print("Name attraction type: ");  
 Ticket.AttractionType category =   
 Ticket.AttractionType.*valueOf*(commandString.nextLine());  
   
 System.*out*.print("Enter ticket cost: ");  
 int cost = commandString.nextInt();  
   
 int day, month, year;  
 System.*out*.print("Type the date (day): ");  
 day = commandString.nextInt();  
 System.*out*.print("Type the date (month): ");  
 month = commandString.nextInt();  
 System.*out*.print("Type the date (year): ");  
 year = commandString.nextInt();  
 p.setSecondName(secondName);  
 p.setCategory(category);  
 p.setCost(cost);  
 p.setDate(day, month, year);  
 }  
 catch (Exception e) {  
 System.*out*.println("Please \"AddNewTicket\""  
 + e.getMessage());  
 }  
 }  
   
 public static void main(String[] args) {  
 ArrayList<Ticket> client = new ArrayList<Ticket>();  
 client.add(new Ticket("Senuk", Ticket.AttractionType.*TRAMPOLINE*, 100, new GregorianCalendar(2021, 3, 23)));  
 client.add(new Ticket("Kotelevich", Ticket.AttractionType.*SHOOTING\_GALLERY*, 200, new GregorianCalendar(2021, 4, 17)));  
 client.add(new Ticket("Larin", Ticket.AttractionType.*FERRIS\_WHEEL*, 500, new GregorianCalendar(2021, 2, 14)));  
 client.add(new Ticket("Nikolov", Ticket.AttractionType.*CARS*, 100, new GregorianCalendar(2021, 2, 15)));  
 client.add(new Ticket("Anonim", Ticket.AttractionType.*FERRIS\_WHEEL*, 0, new GregorianCalendar(2021, 4, 6)));  
   
 System.*out*.println("Type in \"Help\" to display the command list");  
   
 String command = "";  
 Scanner commandString = new Scanner(System.*in*);  
 do {  
 command = commandString.nextLine();  
 switch(command) {  
 case "PrintNames":  
 *printListName*(client);  
 System.*out*.println();  
 break;  
 case "SortByName":  
 client.sort(Ticket.*sortingByName*);  
 *printListCategory*(client);  
 System.*out*.println("sorted successfully");  
 break;  
 case "PrintTicketInfo":  
 *printListCategory*(client);  
 System.*out*.println();  
 break;  
 case "SortByCategory":  
 client.sort(Ticket.*sortingByCategory*);  
 *printListCategory*(client);  
 System.*out*.println("sorted successfully");  
 break;  
  
 case "AddNewTicket":  
 Ticket addPolis = new Ticket();  
 *editTicket*(addPolis);  
 client.add(addPolis);  
 *printListCategory*(client);  
 System.*out*.print ("Successfully added to list!");  
  
 break;  
 case "EditTickets":  
 System.*out*.print("Input second name ");  
 String secondName = commandString.nextLine();  
 for(Ticket p : client) {  
 if (p.getSecondName().equals(secondName)) {  
 *editTicket*(p);  
 *printListCategory*(client);  
 System.*out*.print ("Successfully edited!");  
 break;  
 }  
 }  
 break;  
 case "DeleteTicket":  
 System.*out*.print("Input second name ");  
 String secondNameForDeleting = commandString.nextLine();  
 for(Ticket p : client) {  
 if (p.getSecondName().equals(secondNameForDeleting)) {  
 client.remove(p);  
 *printListCategory*(client);  
 System.*out*.print ("Successfully del!");  
 break;  
 }  
 }  
 break;  
 case "CSVFromTxtLoder":  
 CSVLoader loader = new CSVLoader();  
 System.*out*.println("Type path with format");  
 String c = commandString.nextLine();  
 File f = new File(c);  
 if (f.exists()) {  
 client = loader.loadTickets(c);  
 *printListCategory*(client);  
 System.*out*.println("Loded from CSV successfully");  
 }  
 break;  
 case "CSVToTxtSaver":  
 CSVLoader load = new CSVLoader();  
 System.*out*.println("Type path with format");  
 String s = commandString.nextLine();  
 load.saveTickets(s,client);  
 System.*out*.println("loaded to CSV successfully");  
  
 break;  
  
 case "Help":  
 System.*out*.println("\"PrintNames\" - list of clients ");  
 System.*out*.println("\"PrintTicketInfo\" - prints list of tickets ");  
 System.*out*.println("\"SearchByDate\" - searches all tickets"  
 + "by the given date");  
 System.*out*.println("\"AddNewTicket\" - Adds new ticket to "  
 + "the list");  
 System.*out*.println("\"EditTickets\" - Edits tickets by "  
 + "given name");  
 System.*out*.println("\"DeleteTicket\" - Deletes the ticket "  
 + "by given name");  
 System.*out*.println("\"Exit\" - exit program.");  
 System.*out*.println("\"SortByCategory\" - Sort tickets by "  
 + "categoty");  
 System.*out*.println("\"SortByName\" - Sort tickets by "  
 + "name");  
 System.*out*.println("\"CSVFromTxtLoder\" - load data from csv.");  
 System.*out*.println("\"CSVToTxtSaver\" - load data to csv.");  
 break;  
// default:  
// throw new IllegalStateException("Unexpected value: " + command);  
 }  
 }  
 while(!command.equals("Exit"));  
 System.*out*.println("Have a nice day");  
   
 }  
}

package com.AttractionsMaven;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.GregorianCalendar;  
import java.util.Comparator;  
  
public class Ticket {  
 private String secondName;  
 private int cost;  
 public void setDate(Date ticketDate) {  
 }  
  
 public enum AttractionType {  
 *CARS*, *TRAMPOLINE*, *FERRIS\_WHEEL*, *ROOM\_OF\_FEAR*, *SHOOTING\_GALLERY*, *CAROUSEL*;  
 }  
 AttractionType category;  
 GregorianCalendar ticketDate = new GregorianCalendar();  
   
 public static Comparator<Ticket> *sortingByName* =   
 new Comparator<Ticket>() {  
 @Override  
 public int compare(Ticket ticket,   
 Ticket ticket2) {  
 return ticket.secondName.compareTo(ticket2.secondName);  
 }  
 };  
   
 public static Comparator<Ticket> *sortingByCategory* =   
 new Comparator<Ticket>() {  
 @Override  
 public int compare(Ticket category1,   
 Ticket category2) {  
 return category1.category.toString().compareTo  
 (category2.category.toString());  
 }  
 };  
   
 public Ticket(String secondName, AttractionType category, int cost,   
 GregorianCalendar ticketDate) {  
 this.secondName = secondName;  
 this.cost = cost;  
 this.category = category;  
 this.ticketDate = ticketDate;  
 }  
  
 public Ticket() {  
 this.secondName = "";  
 this.cost = 0;  
 this.category = AttractionType.*CAROUSEL*;  
 this.ticketDate = new GregorianCalendar();  
 }  
  
 public String getSecondName() {  
 return secondName;  
 }  
  
 public int getCost() {  
 return cost;  
 }  
  
 public AttractionType getCategory() {  
 return category;  
 }  
   
 public String getDate() {  
 return ticketDate.get(Calendar.*DAY\_OF\_MONTH*) + "."   
 + ticketDate.get(Calendar.*MONTH*) + "."   
 + ticketDate.get(Calendar.*YEAR*);  
 }  
 public Calendar getCalendar (){  
 return ticketDate;  
 }  
  
 public void setSecondName(String secondName) {  
 this.secondName = secondName;  
 }  
  
 public void setCost(int cost) {  
 this.cost = cost;  
 }  
  
 public void setCategory(AttractionType category) {  
 this.category = category;  
 }  
  
 public void setDate(int day, int month, int year) {  
 ticketDate.set(year, month, day);  
 }  
   
   
}

package com.AttractionsMaven;  
  
import com.opencsv.CSVReader;  
import com.opencsv.CSVWriter;  
  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.GregorianCalendar;  
  
public class CSVLoader {  
  
 public ArrayList<Ticket> loadTickets(String sourceName) {  
 ArrayList<Ticket> polisesList = new ArrayList<>();  
 try {  
 FileReader filereader = new FileReader(sourceName);  
  
 CSVReader reader = new CSVReader(filereader);  
 DateFormat df = new SimpleDateFormat("dd.MM.yyyy");  
 String[] nextLine;  
  
 while ((nextLine = reader.readNext()) != null) {  
 String secondName = nextLine[0];  
 Ticket.AttractionType type = Ticket.AttractionType.*valueOf* (nextLine[1]);  
 int cost = Integer.*parseInt*(nextLine[2]);  
  
 String ticketExtradition = nextLine[3];  
 GregorianCalendar iaCalendar = new GregorianCalendar();  
 Date ticketDate = df.parse(ticketExtradition);  
 iaCalendar.setGregorianChange(ticketDate);  
  
 Ticket ticket = new Ticket(secondName, type, cost,  
 iaCalendar);  
 polisesList.add(ticket);  
  
 }  
 } catch (Exception e) {  
 System.*err*.println("Error occured while loading " + e.getMessage());  
 return null;  
 }  
 return polisesList;  
 }  
  
 public void saveTickets(String sourceName, ArrayList<Ticket> client) {  
 try {  
 CSVWriter writer = new CSVWriter(new FileWriter(sourceName));  
  
// DateFormat df = new SimpleDateFormat("dd.MM.yyyy");  
   
 for (Ticket ticket : client) {  
 String[] entity = new String[4];  
 entity[0] = ticket.getSecondName();  
 entity[1] = ticket.getCategory().toString();  
 entity[2] = Integer.*toString*(ticket.getCost());  
 entity[3] = ticket.getDate();  
  
 writer.writeNext(entity);  
 }  
 writer.close();  
 }   
 catch (Exception m) {  
 System.*err*.println("Error occured while saving " + m.getMessage());  
 }  
 }  
   
}