**МИНИСТЕРСТВО**

**ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
 УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине: «**Визуальные средства разработки программных приложений»**

на тему: «Обработка данных с использованием *XML*»

Выполнил: студент гр. ИТП-31

Расшивалов Н.И.  
 Принял: ассистент

Михалевич В.Г.

Гомель 2022

**Цель работы:** изучить методы обработки данных с использованием *XML java*.

Задание представлено на рисунке 1.

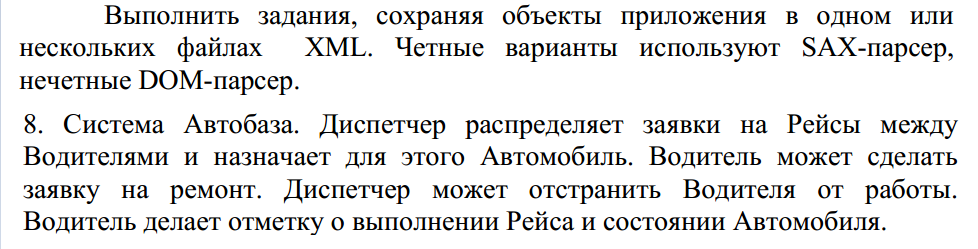


Рисунок 1 – Вариант задания

Листинг программы представлен в приложении А.

Скриншоты выполнения работы представлены в приложении Б.

**Вывод:** в ходе выполнения лабораторной работы, были изучены потоки ввода-вывода с использованием пакета *java.io*.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

package com.company;  
  
import com.company.data.CarRepository;  
import com.company.data.PersonRepository;  
import com.company.data.TripRepository;  
import services.BaseService;  
  
import java.io.IOException;  
  
public class Main {  
 private static String personsStoragePath = System.getProperty("user.dir") + "\\files\\persons.json";  
 private static String tripsStoragePath = System.getProperty("user.dir") + "\\files\\trips.json";  
 private static String carsStoragePath = System.getProperty("user.dir") + "\\files\\cars.json";  
  
  
 public static void main(String[] args) throws IOException {  
 var personRepository = new PersonRepository(personsStoragePath);  
 var tripsRepository = new TripRepository(tripsStoragePath);  
 var carRepository = new CarRepository(carsStoragePath);  
 var service = new BaseService(personRepository, tripsRepository,carRepository);  
 service.ShowMenu();  
 }  
}

package com.company.data;  
  
import models.Entity;  
  
import java.util.List;  
  
public interface IRepository<T extends Entity> {  
 int Create(T entity);  
  
 boolean Update(T entity);  
  
 List<T> Get();  
  
 default int GetIndex(List<T> items, int id) {  
 for (int i = 0; i < items.size(); i++) {  
 if (id == items.get(i).getId()) {  
 return i;  
 }  
 }  
  
 return -1;  
 }  
}

package com.company.data;  
  
import com.fasterxml.jackson.annotation.JsonAutoDetect;  
import com.fasterxml.jackson.annotation.PropertyAccessor;  
import com.fasterxml.jackson.core.type.TypeReference;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import models.Car;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
public class CarRepository implements IRepository<Car>  
{  
 private String \_path;  
  
 public CarRepository(String path) {  
 \_path = path;  
 }  
  
 public int Create(Car entity) {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 var file = new File(\_path);  
 try {  
 int id;  
 List<Car> cars = new ArrayList<>();  
 if (file.exists()) {  
 cars = mapper.readValue(file, new TypeReference<List<Car>>() {  
 });  
 id = cars.stream().count() > 0 ? cars.stream().max(Comparator.comparing(Car::getId)).get().getId() + 1 : 1;  
 } else {  
 id = 1;  
 }  
 entity.setId(id);  
 cars.add(entity);  
 mapper.writeValue(file, cars);  
 return id;  
 } catch (Exception e) {  
 }  
 return -1;  
 }  
  
 public boolean Update(Car entity) {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 var file = new File(\_path);  
 try {  
 var cars = mapper.readValue(file, new TypeReference<List<Car>>() {  
 });  
 var index = GetIndex(cars, entity.getId());  
 if (index > 0) {  
 cars.remove(index);  
 cars.add(index, entity);  
 mapper.writeValue(file, cars);  
 return true;  
 }  
 } catch (Exception e) {  
 }  
 return false;  
 }  
  
 public List<Car> Get() {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 try {  
 var file = new File(\_path);  
 return mapper.readValue(file, new TypeReference<List<Car>>() {  
 });  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
  
 return new ArrayList<>();  
 }  
}

package com.company.data;  
  
import com.fasterxml.jackson.annotation.JsonAutoDetect;  
import com.fasterxml.jackson.annotation.PropertyAccessor;  
import com.fasterxml.jackson.core.type.TypeReference;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import models.Person;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
public class PersonRepository implements IRepository<Person> {  
  
 private String \_path;  
  
 public PersonRepository(String path) {  
 \_path = path;  
 }  
  
 public int Create(Person entity) {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 var file = new File(\_path);  
 try {  
 int id;  
 List<Person> persons = new ArrayList<>();  
 if (file.exists()) {  
 persons = mapper.readValue(file, new TypeReference<List<Person>>() {  
 });  
 id = persons.stream().count() > 0 ? persons.stream().max(Comparator.comparing(Person::getId)).get().getId() + 1 : 1;  
 } else {  
 id = 1;  
 }  
 entity.setId(id);  
 persons.add(entity);  
 mapper.writeValue(file, persons);  
 return id;  
 } catch (Exception e) {  
 }  
 return -1;  
 }  
  
 public boolean Update(Person entity) {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 var file = new File(\_path);  
 try {  
 var persons = mapper.readValue(file, new TypeReference<List<Person>>() {  
 });  
 var index = GetIndex(persons, entity.getId());  
 if (index > 0) {  
 persons.remove(index);  
 persons.add(index, entity);  
 mapper.writeValue(file, persons);  
 return true;  
 }  
 } catch (Exception e) {  
 }  
 return false;  
 }  
  
 public List<Person> Get() {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 try {  
 var file = new File(\_path);  
 return mapper.readValue(file, new TypeReference<List<Person>>() {  
 });  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
  
 return new ArrayList<>();  
 }  
  
}

package com.company.data;  
  
import com.fasterxml.jackson.annotation.JsonAutoDetect;  
import com.fasterxml.jackson.annotation.PropertyAccessor;  
import com.fasterxml.jackson.core.type.TypeReference;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import models.Trip;  
  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
public class TripRepository implements IRepository<Trip> {  
 private String \_path;  
  
 public TripRepository(String path) {  
 \_path = path;  
 }  
  
 public int Create(Trip entity) {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 var file = new File(\_path);  
 try {  
 int id;  
 List<Trip> trips = new ArrayList<>();  
 if (file.exists()) {  
 trips = mapper.readValue(file, new TypeReference<List<Trip>>() {  
 });  
 id = trips.stream().count() > 0 ? trips.stream().max(Comparator.comparing(Trip::getId)).get().getId() + 1 : 1;  
 } else {  
 id = 1;  
 }  
 entity.setId(id);  
 trips.add(entity);  
 mapper.writeValue(file, trips);  
 return id;  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return -1;  
 }  
  
 public boolean Update(Trip entity) {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 var file = new File(\_path);  
 try {  
 var trips = mapper.readValue(file, new TypeReference<List<Trip>>() {  
 });  
 var index = GetIndex(trips, entity.getId());  
 if (index >= 0) {  
 trips.remove(index);  
 trips.add(index, entity);  
 mapper.writeValue(file, trips);  
 return true;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 System.out.println(e.getMessage());  
 }  
 return false;  
 }  
  
 public List<Trip> Get() {  
 var mapper = new ObjectMapper();  
 mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);  
 try {  
 var file = new File(\_path);  
 return mapper.readValue(file, new TypeReference<List<Trip>>() {  
 });  
 } catch (IOException e) {  
 }  
  
 return new ArrayList<>();  
 }  
}

package services;  
  
import com.company.data.CarRepository;  
import com.company.data.PersonRepository;  
import com.company.data.TripRepository;  
import models.\*;  
  
import java.util.Arrays;  
  
public class AdminService extends BaseService {  
 private Person \_currentPerson;  
  
 public AdminService  
 (  
 Person currentPerson,  
 PersonRepository personRepository,  
 TripRepository tripRepository,  
 CarRepository carRepository  
 ) {  
 super(personRepository, tripRepository, carRepository);  
 \_currentPerson = currentPerson;  
 }  
  
 @Override  
 public void ShowMenu() {  
 while (true) {  
 var choice = GetChoice(Arrays.asList("1.Add dispatcher", "2.Add driver", "3.Show users", "4.Add car","5.Exit"), 5);  
 switch (choice) {  
 case 1:  
 AddDispatcher();  
 break;  
 case 2:  
 AddDriver();  
 break;  
 case 3:  
 ShowUsers();  
 break;  
 case 4:  
 AddCar();  
 case 5:return;  
 default:  
 break;  
 }  
 }  
 }  
  
 public void AddDispatcher() {  
 System.out.println("Input username:");  
 String username = \_scanner.nextLine();  
 if (!CheckUsername(username)) {  
 return;  
 }  
 System.out.println("Input password:");  
 String password = \_scanner.nextLine();  
 System.out.println("Input name:");  
 String name = \_scanner.nextLine();  
 System.out.println("Input surname:");  
 String surname = \_scanner.nextLine();  
 var dispatcher = new Dispatcher(username, password, name, surname);  
 \_personRepository.Create(dispatcher);  
 }  
  
 public void AddCar() {  
 System.out.println("Input stamp:");  
 String stamp = \_scanner.nextLine();  
  
 System.out.println("Input number:");  
 String number = \_scanner.nextLine();  
  
 System.out.println("Input isBroken:");  
 Boolean isBroken = false;  
  
 System.out.println("Choose car type:\n1 - Tank truck; 2 - Refrigerator; 3 - Container ship ");  
 int value = \_scanner.nextInt();  
 switch (value) {  
 case 1:  
 \_carRepository.Create(new TankTruck(stamp, number, isBroken));  
 break;  
 case 2:  
 \_carRepository.Create(new Refrigerator(stamp, number, isBroken));  
 break;  
 case 3:  
 \_carRepository.Create(new ContainerShip(stamp, number, isBroken));  
 break;  
 default:  
 return;  
 }  
 }  
  
  
 public void AddDriver() {  
 System.out.println("Input username:");  
 String username = \_scanner.nextLine();  
 if (!CheckUsername(username)) {  
 return;  
 }  
 System.out.println("Input password:");  
 String password = \_scanner.nextLine();  
 System.out.println("Input name:");  
 String name = \_scanner.nextLine();  
 System.out.println("Input surname:");  
 String surname = \_scanner.nextLine();  
 var driver = new Driver(username, password, name, surname);  
 \_personRepository.Create(driver);  
 }  
  
 public void ShowUsers() {  
 int i = 1;  
 for (Person person : \_personRepository.Get()) {  
 System.out.println(i + ". " + person.toString());  
 i++;  
 }  
 }  
  
  
 private boolean CheckUsername(String username) {  
 if (username == null) {  
 System.out.println("Incorrect username");  
 return false;  
 }  
 var existPerson = \_personRepository.Get().stream().filter(p -> p.getUsername().equals(username)).findFirst();  
 if (existPerson.isPresent()) {  
 System.out.println("Such user already exists");  
 return false;  
 }  
  
 return true;  
 }  
  
  
}

package services;  
  
import com.company.data.CarRepository;  
import com.company.data.PersonRepository;  
import com.company.data.TripRepository;  
import models.Person;  
  
import java.util.Arrays;  
import java.util.List;  
import java.util.Optional;  
import java.util.Scanner;  
  
public class BaseService {  
 protected PersonRepository \_personRepository;  
 protected TripRepository \_tripRepository;  
 protected CarRepository \_carRepository;  
  
 protected Scanner \_scanner = new Scanner(System.in);  
  
 public BaseService(PersonRepository personRepository, TripRepository tripRepository, CarRepository carRepository) {  
 \_personRepository = personRepository;  
 \_tripRepository = tripRepository;  
 \_carRepository = carRepository;  
 }  
  
 public void ShowMenu() {  
 while (true) {  
 var choice = GetChoice(Arrays.asList("1.Login", "2.Exit"), 2);  
 switch (choice) {  
 case 1:  
 var optionalPerson = login();  
 if (optionalPerson.isPresent()) {  
 var person = optionalPerson.get();  
 var type = person.getType();  
 switch (type) {  
 case Admin -> new AdminService(person, \_personRepository, \_tripRepository, \_carRepository).ShowMenu();  
 case Dispatcher -> new DispatcherService(person, \_personRepository, \_tripRepository, \_carRepository).ShowMenu();  
 case Driver -> new DriverService(person, \_personRepository, \_tripRepository, \_carRepository).ShowMenu();  
 }  
 }  
 break;  
 case 2:  
 return;  
 default:  
 break;  
 }  
 }  
 }  
  
 private Optional<Person> login() {  
 System.out.println("Input username:");  
 String username = \_scanner.nextLine();  
 var persons = \_personRepository.Get();  
 var person = persons.stream().filter(p -> p.getUsername().equals(username)).findFirst();  
 if (person.isEmpty()) {  
 System.out.println("No such person");  
 return Optional.empty();  
 }  
  
 System.out.println("Input password:");  
 String password = \_scanner.nextLine();  
 if (!person.get().getPassword().equals(password)) {  
 System.out.println("Incorrect password");  
 return Optional.empty();  
 }  
  
 return person;  
 }  
  
  
 public int GetChoice(List<String> menuItems, int maxInput) {  
 System.out.println("Menu");  
 for (String item :  
 menuItems) {  
 System.out.println(item);  
 }  
 System.out.println("Select:");  
 int choice;  
 while (true) {  
 try {  
 choice = Integer.parseInt(\_scanner.nextLine());  
 if (choice < 1 || choice > maxInput) throw new Exception("");  
 break;  
 } catch (Exception exception) {  
 System.out.println("Incorrect input");  
 }  
 }  
 return choice;  
 }  
}

package services;  
  
import com.company.data.CarRepository;  
import com.company.data.PersonRepository;  
import com.company.data.TripRepository;  
import models.Car;  
import models.Driver;  
import models.Person;  
import models.Trip;  
  
import java.text.SimpleDateFormat;  
import java.util.Arrays;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class DispatcherService extends BaseService {  
 private Person \_currentPerson;  
  
 public DispatcherService  
 (  
 Person currentPerson,  
 PersonRepository personRepository,  
 TripRepository tripRepository,  
 CarRepository carRepository  
 ) {  
 super(personRepository, tripRepository, carRepository);  
 \_currentPerson = currentPerson;  
 }  
  
 @Override  
 public void ShowMenu() {  
 while (true) {  
 var choice = GetChoice(Arrays.asList("1.Add trip", "2.Suspend the driver", "3.Exit"), 4);  
 switch (choice) {  
 case 1:  
 AddTrip();  
 break;  
 case 2:  
 SetIsWorked();  
 break;  
 case 3:  
 return;  
 default:  
 break;  
 }  
 }  
 }  
  
 public void AddTrip() {  
 var drivers = \_personRepository.Get().stream().filter(t -> t.getClassName().equals("models.Driver")).collect(Collectors.toList());  
 var driverId = GetDriverId(drivers);  
 var cars = \_carRepository.Get().stream().collect(Collectors.toList());  
 var carId = GetCarId(cars);  
 Calendar calendar = Calendar.getInstance();  
 SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");  
 var trip = new Trip(carId, formatter.format(calendar.getTime()), driverId);  
 trip.set\_isCompleted(false);  
 \_tripRepository.Create(trip);  
 }  
  
 public void SetIsWorked() {  
 try {  
 var drivers = \_personRepository.Get().stream().filter(t -> t.getClassName().equals("models.Driver")).collect(Collectors.toList());  
 var driverId = GetDriverId(drivers);  
 var driver = \_personRepository.Get().stream().filter(t -> t.getId() == driverId).findFirst();  
 if (driver.isEmpty()) {  
 throw new Exception("");  
 }  
 var currentDriver = (Driver) driver.get();  
 if (currentDriver.get\_isWorked() == false) {  
 currentDriver.set\_isWorked(true);  
 } else {  
 currentDriver.set\_isWorked(false);  
 }  
 \_personRepository.Update(driver.get());  
 } catch (Exception exception) {  
 System.out.println("Incorrect driver id");  
 return;  
 }  
 }  
  
 private int GetCarId(List<Car> items) {  
 for (Car car : items) {  
 System.out.println(car.getId() + ". " + car.toString());  
 }  
 System.out.println("Select:");  
 try {  
 var id = Integer.parseInt(\_scanner.nextLine());  
 if (items.stream().filter(p -> p.getId() == id).count() == 0) {  
 throw new Exception("");  
 }  
 return id;  
 } catch (Exception exception) {  
 System.out.println("Incorrect person id");  
 }  
  
 return -1;  
 }  
  
 private int GetDriverId(List<Person> items) {  
 for (Person driver : items) {  
 System.out.println(driver.getId() + ". " + driver.toString());  
 }  
 System.out.println("Select:");  
 try {  
 var id = Integer.parseInt(\_scanner.nextLine());  
 if (items.stream().filter(p -> p.getId() == id).count() == 0) {  
 throw new Exception("");  
 }  
 return id;  
 } catch (Exception exception) {  
 System.out.println("Incorrect person id");  
 }  
  
 return -1;  
 }  
}

package services;  
  
import com.company.data.CarRepository;  
import com.company.data.PersonRepository;  
import com.company.data.TripRepository;  
import models.\*;  
  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
  
  
public class DriverService extends BaseService {  
 private final Person \_currentPerson;  
  
 public DriverService  
 (  
 Person currentPerson,  
 PersonRepository personRepository,  
 TripRepository tripRepository,  
 CarRepository carRepository  
 ) {  
 super(personRepository, tripRepository, carRepository);  
 \_currentPerson = currentPerson;  
 }  
  
 @Override  
 public void ShowMenu() {  
 while (true) {  
 var choice = GetChoice(Arrays.asList("1.Show information", "2.Exit"), 2);  
 switch (choice) {  
 case 1:  
 Driver driver = (Driver) ShowInfo();  
 var value = GetChoice(Arrays.asList("1.Repair request", "2.Make trip status", "3.Exit"), 3);  
 switch (value) {  
 case 1:  
 RepairRequest(driver);  
 break;  
 case 2:  
 MakeStatus(driver);  
 break;  
 case 3:  
 return;  
 default:  
 break;  
 }  
 break;  
 case 2:  
 return;  
 default:  
 break;  
 }  
 }  
 }  
  
 public Person ShowInfo() {  
 try {  
 var driverId = \_currentPerson.getId();  
 var driver = \_personRepository.Get().stream().filter(t -> t.getId() == driverId).findFirst();  
 if (driver.isEmpty()) {  
 throw new Exception("");  
 }  
 System.out.println(driver.get().toString());  
 return driver.get();  
 } catch (Exception exception) {  
 System.out.println("Incorrect driver id");  
 return null;  
 }  
 }  
  
 public void RepairRequest(Driver driver) {  
 try {  
 var car = \_carRepository.Get().stream().filter(t -> t.getId() == driver.get\_carId()).findFirst().get();  
 System.out.println(car.toString());  
 if (car.is\_isBroken() == true) {  
 car.set\_isBroken(false);  
 }  
 System.out.println("Complete");  
 System.out.println(car.toString());  
 \_carRepository.Update(car);  
 } catch (Exception e) {  
 System.out.println(e.getMessage());  
 return;  
 }  
 }  
  
 public void MakeStatus(Driver driver) {  
 try {  
 var car = \_carRepository.Get().stream().filter(t -> t.getId() == driver.get\_carId()).findFirst().get();  
 System.out.println(car);  
 var trips = \_tripRepository.Get().stream().collect(Collectors.toList());  
 var tripId = GetTripId(trips);  
 var trip = \_tripRepository.Get().stream().filter(t -> t.getId() == tripId).findFirst().get();  
 System.out.println("Choose status 1-complete or 2 - non-complete");  
 var value = Integer.parseInt(\_scanner.nextLine());  
 switch (value) {  
 case 1:  
 trip.set\_isCompleted(true);  
 MakeCarStatus(car);  
 break;  
 case 2:  
 trip.set\_isCompleted(false);  
 MakeCarStatus(car);  
 break;  
 default:  
 System.out.println("Input error");  
 return;  
 }  
 \_carRepository.Update(car);  
 \_tripRepository.Update(trip);  
 } catch (Exception e) {  
 System.out.println(e.getMessage());  
 return;  
 }  
 }  
  
 public void MakeCarStatus(Car car) {  
 System.out.println("Choose status 1-not broken or 2 - broken");  
 var value = Integer.parseInt(\_scanner.nextLine());  
 switch (value) {  
 case 1:  
 car.set\_isBroken(true);  
 break;  
 case 2:  
 car.set\_isBroken(false);  
 break;  
 default:  
 System.out.println("Input error");  
 return;  
 }  
 }  
  
 private int GetTripId(List<Trip> items) {  
 for (Trip trip : items) {  
 System.out.println(trip.getId() + ". " + trip.toString());  
 }  
 System.out.println("Select:");  
 try {  
 var id = Integer.parseInt(\_scanner.nextLine());  
 if (items.stream().filter(p -> p.getId() == id).count() == 0) {  
 throw new Exception("");  
 }  
 return id;  
 } catch (Exception exception) {  
 System.out.println("Incorrect person id");  
 }  
  
 return -1;  
 }  
}

package models;  
  
  
public class Admin extends Person {  
 public Admin() {  
 }  
  
 public Admin(String username, String password, String name, String surname) {  
 super(username, password, name, surname);  
 }  
  
 @Override  
 public PersonType getType() {  
 return PersonType.Admin;  
 }  
}

package models;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
  
public abstract class Car extends Entity {  
 private String \_stamp;  
 private String \_number;  
 private boolean \_isBroken;  
 private CarType \_type;  
  
  
 public Car() {  
 }  
  
 public Car(String stamp, String number, Boolean isBroken,CarType type) {  
 \_stamp = stamp;  
 \_number = number;  
 \_isBroken = isBroken;  
 \_type = type;  
 }  
  
 @JsonIgnore  
 public abstract CarType getType();  
  
 public String get\_stamp() {  
 return \_stamp;  
 }  
  
 public void set\_stamp(String \_stamp) {  
 this.\_stamp = \_stamp;  
 }  
  
 public String get\_number() {  
 return \_number;  
 }  
  
 public void set\_number(String \_number) {  
 this.\_number = \_number;  
 }  
  
 public boolean is\_isBroken() {  
 return \_isBroken;  
 }  
  
 public void set\_isBroken(boolean \_isBroken) {  
 this.\_isBroken = \_isBroken;  
 }  
  
 @Override  
 public String toString() {  
  
 return \_type + ", " + \_stamp + ", " + \_number + ", " + \_isBroken;  
 }  
}

package models;  
  
public enum CarType {  
 ContainerShip,  
 Refrigerator,  
 TankTruck;  
  
 public static CarType fromInteger(int x) {  
 switch (x) {  
 case 1:  
 return ContainerShip;  
 case 2:  
 return Refrigerator;  
 case 3:  
 return TankTruck;  
 }  
 return null;  
 }  
}

package models;  
  
public class ContainerShip extends Car {  
 private int \_daysNumber;  
  
 public ContainerShip() {  
 }  
  
 public ContainerShip(String stamp, String number, Boolean isBroken) {  
 super(stamp, number, isBroken, CarType.ContainerShip);  
  
 }  
  
  
 @Override  
 public CarType getType() {  
 return CarType.ContainerShip;  
 }  
}

package models;  
  
public class Dispatcher extends Person {  
 public Dispatcher() {  
 }  
  
 public Dispatcher(String username, String password, String name, String surname) {  
 super(username, password, name, surname);  
 }  
  
 @Override  
 public PersonType getType() {  
 return PersonType.Dispatcher;  
 }  
  
}

package models;  
  
public class Driver extends Person {  
 private Boolean \_isWorked;  
 private int \_tripId;  
 private int \_carId;  
 public Driver() {  
 }  
  
 public Driver(String username, String password, String name, String surname) {  
 super(username, password, name, surname);  
 }  
  
 public Boolean get\_isWorked() {  
 return \_isWorked;  
 }  
  
 public void set\_isWorked(Boolean \_isWorked) {  
 this.\_isWorked = \_isWorked;  
 }  
  
 public int get\_carId() {  
 return \_carId;  
 }  
  
 public void set\_carId(int \_carId) {  
 this.\_carId = \_carId;  
 }  
  
 public int get\_tripId() {  
 return \_tripId;  
 }  
  
 public void set\_tripId(int \_tripId) {  
 this.\_tripId = \_tripId;  
 }  
  
 @Override  
 public String toString() {  
 return super.toString() + ", \_isWorked: " + \_isWorked + ", \_tripId: "+ \_tripId+ ", \_carId: "+ \_carId ;  
 }  
  
 @Override  
 public PersonType getType() {  
 return PersonType.Driver;  
 }  
  
}

package models;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import com.fasterxml.jackson.annotation.JsonProperty;  
import com.fasterxml.jackson.annotation.JsonTypeInfo;  
  
@JsonTypeInfo(  
 use = JsonTypeInfo.Id.CLASS,  
 include = JsonTypeInfo.As.EXISTING\_PROPERTY,  
 property = "@class")  
public abstract class Entity {  
 private int \_id;  
  
 @JsonIgnore  
 public int getId() {  
 return \_id;  
 }  
  
 public void setId(int id) {  
 \_id = id;  
 }  
  
 @JsonProperty("@class")  
 public String getClassName() {  
 return this.getClass().getName();  
 }  
}

package models;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
  
public abstract class Person extends Entity {  
 private String \_name;  
 private String \_surname;  
 private String \_username;  
 private String \_password;  
  
 public Person() {  
 }  
  
 public Person(String username, String password, String name, String surname) {  
 \_username = username;  
 \_password = password;  
 \_name = name;  
 \_surname = surname;  
 }  
  
 @JsonIgnore  
 public abstract PersonType getType();  
  
 @JsonIgnore  
 public String getName() {  
 return \_name;  
 }  
  
 public void setName(String name) {  
 \_name = name;  
 }  
  
 @JsonIgnore  
 public String getSurname() {  
 return \_surname;  
 }  
  
 public void setSurname(String surname) {  
 \_surname = surname;  
 }  
  
 @JsonIgnore  
 public String getUsername() {  
 return \_username;  
 }  
 @JsonIgnore  
 public void setUsername(String username) {  
 \_username = username;  
 }  
  
 @JsonIgnore  
 public String getPassword() {  
 return \_password;  
 }  
 @JsonIgnore  
 public void setPassword(String password) {  
 \_password = password;  
 }  
  
 @Override  
 public String toString() {  
 return getType() + ", " + \_name + " " + \_surname;  
 }  
}

package models;  
  
public enum PersonType {  
 Driver,  
 Dispatcher,  
 Admin,  
}

package models;  
  
public class Refrigerator extends Car {  
  
 public Refrigerator() {  
 }  
  
 public Refrigerator(String stamp, String number, Boolean isBroken) {  
 super(stamp, number, isBroken,CarType.Refrigerator);  
  
 }  
  
  
 @Override  
 public CarType getType() {  
 return CarType.Refrigerator;  
 }  
}

package models;  
  
public class RepairOrder {  
 private int \_carId;  
 private int \_driverId;  
  
 public RepairOrder(int carId, int driverId) {  
 this.\_carId = carId;  
 this.\_driverId = driverId;  
 }  
 public int get\_carId() {  
 return \_carId;  
 }  
  
 public void set\_carId(int \_carId) {  
 this.\_carId = \_carId;  
 }  
  
 public int get\_driverId() {  
 return \_driverId;  
 }  
  
 public void set\_driverId(int \_driverId) {  
 this.\_driverId = \_driverId;  
 }  
  
  
}

package models;  
  
public class TankTruck extends Car {  
  
 public TankTruck(){}  
 public TankTruck(String stamp, String number, Boolean isBroken) {  
 super(stamp, number, isBroken,CarType.TankTruck);  
  
 }  
  
  
 @Override  
 public CarType getType() {  
 return CarType.TankTruck;  
 }  
}

package models;  
  
public class Trip extends Entity {  
 private Boolean \_isCompleted;  
 private String \_date;  
 private int \_driverId;  
 private int \_carId;  
  
 public Trip(){}  
 public Trip(int carId,String date,int \_driverId) {  
  
 this.\_date = date;  
 this.\_carId=carId;  
 this.\_driverId = \_driverId;  
 }  
 public int get\_carId() {  
 return \_carId;  
 }  
  
 public void set\_carId(int \_carId) {  
 this.\_carId = \_carId;  
 }  
  
 public int get\_driverId() {  
 return \_driverId;  
 }  
  
 public void set\_driverId(int \_driverId) {  
 this.\_driverId = \_driverId;  
 }  
  
 public String get\_date() {  
 return \_date;  
 }  
  
 public void set\_date(String \_date) {  
 this.\_date = \_date;  
 }  
  
 public Boolean get\_isCompleted() {  
 return \_isCompleted;  
 }  
  
 public void set\_isCompleted(Boolean \_isCompleted) {  
 this.\_isCompleted = \_isCompleted;  
 }  
  
 @Override  
 public String toString() {  
 return \_date + ", " + \_carId + ", " + \_driverId;  
 }  
}

package Utils;  
  
import models.Car;  
import models.CarType;  
import org.xml.sax.Attributes;  
import org.xml.sax.SAXException;  
import org.xml.sax.helpers.DefaultHandler;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class CarHandler extends DefaultHandler {  
  
 private boolean hasStamp = false;  
 private boolean hasNumber = false;  
 private boolean hasType = false;  
 private boolean hasIsBroken = false;  
  
 // List to hold Car object  
 private List< Car > carList = null;  
 private Car car = null;  
  
 public List <Car> getEmpList() {  
 return carList;  
 }  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  
  
 if (qName.equalsIgnoreCase("Car")) {  
 String id = attributes.getValue("id");  
 // initialize Car object and set id attribute  
 car = new Car() {  
 @Override  
 public CarType getType() {  
 return null;  
 }  
 };  
 car.setId(Integer.parseInt(id));  
 // initialize list  
 if (carList == null)  
 carList = new ArrayList< >();  
 } else if (qName.equalsIgnoreCase("\_stamp")) {  
 // set boolean values for fields, will be used in setting Car variables  
 hasStamp = true;  
 } else if (qName.equalsIgnoreCase("\_number")) {  
 hasNumber = true;  
 } else if (qName.equalsIgnoreCase("\_isBroken")) {  
 hasIsBroken = true;  
 } else if (qName.equalsIgnoreCase("\_type")) {  
 hasType = true;  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) throws SAXException {  
 if (qName.equalsIgnoreCase("Car")) {  
 // add User object to list  
 carList.add(car);  
 }  
 }  
  
 @Override  
 public void characters(char ch[], int start, int length) throws SAXException {  
  
 if (hasStamp) {  
 car.set\_stamp((new String(ch, start, length)));  
 hasStamp = false;  
 } else if (hasNumber) {  
 car.set\_number(new String(ch, start, length));  
 hasNumber = false;  
 } else if (hasType) {  
 car.set\_type(CarType.valueOf(new String(ch, start, length)));  
 hasType = false;  
 } else if (hasIsBroken) {  
 car.set\_isBroken(false);  
 hasIsBroken = false;  
 }  
 }  
}

**ПРИЛОЖЕНИЕ Б**

**Скриншоты программы**

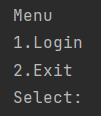


Рисунок 2 – Меню авторизации

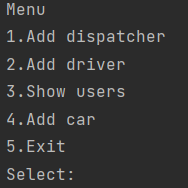


Рисунок 3 – Меню администратора

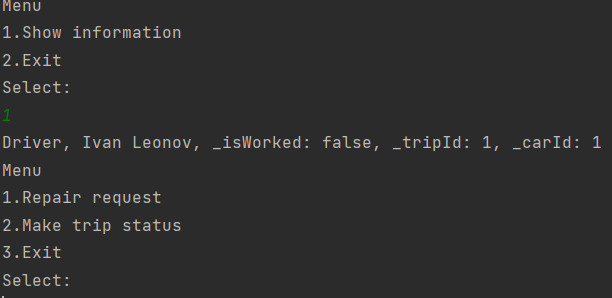


Рисунок 4 – Меню водителя

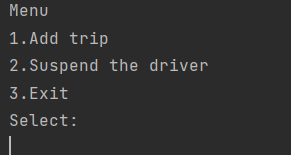


Рисунок 4 – Меню диспетчера



Рисунок 6 – Сериализованные данные автомобилей

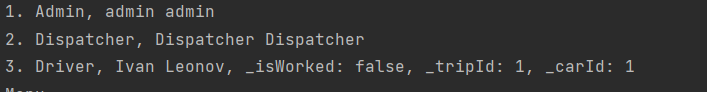


Рисунок 6 – Вывод информации о пользователях