Dawid Waligórski, 264015
Michał Dziedziak, 263901

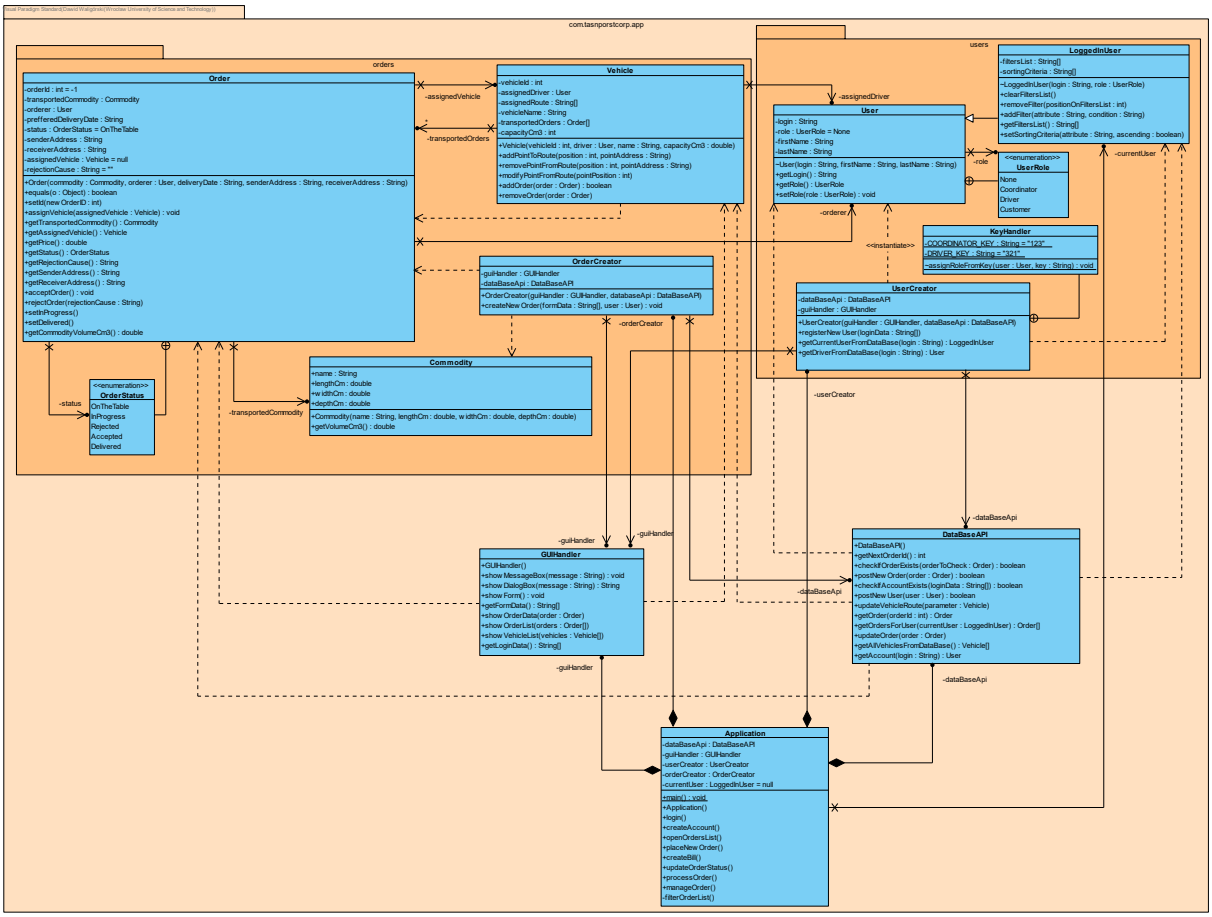
Laboratorium 7, 8, 9, 10

- Identyfikacja klas reprezentujących logikę biznesową projektowanego oprogramowania. Definicja atrybutów i operacji klas oraz związków między klasami na podstawie analizy scenariuszy przypadków użycia. Opracowanie diagramów klas i pakietów. Zastosowanie projektowych wzorców strukturalnych i wytwórczych.
- Opracowanie diagramów sekwencji dla wybranych przypadków użycia, reprezentujących usługi oprogramowania, wynikających również z wykonanych diagramów czynności. Definicja operacji klas na podstawie diagramów sekwencji w języku Java. Zastosowanie projektowych wzorców zachowania.
- Opracowanie diagramów sekwencji dla wybranych przypadków użycia reprezentujących usługi oprogramowania wynikających również z wykonanych diagramów czynności. Definicja operacji klas na podstawie diagramów sekwencji w języku Java. Zastosowanie projektowych wzorców zachowania.

Temat projektu

Program obsługujący zakład transportowy.

Diagram klas:



Kod klas

Application

```
package com.tasnporstcorp.app;

import java.util.*;
import com.tasnporstcorp.app.users.*;
import com.tasnporstcorp.app.orders.*;

public class Application {

	private DataBaseAPI dataBaseApi;
	private GUIHandler guiHandler;
	private UserCreator userCreator;
	private OrderCreator orderCreator;
	private LoggedInUser currentUser = null;

	public static void main(String[] args) {

		Application app = new Application();
		app.createAccount();
		app.login();
		app.placeNewOrder();
```

```
	}

	public Application() {
		guiHandler = new GUIHandler();
		dataBaseApi = new DataBaseAPI();
		orderCreator = new OrderCreator(guiHandler, dataBaseApi);
		userCreator = new UserCreator(guiHandler, dataBaseApi);
	}

	public void login() {
		currentUser = userCreator.getCurrentUserFromDataBase("login");
	}

	public void createAccount() {
		ArrayList<String> loginData = guiHandler.getLoginData();
		userCreator.registerNewUser(loginData);
	}

	public void openOrdersList() {
		throw new UnsupportedOperationException();
	}

	public void placeNewOrder() {
		ArrayList<String> formData = guiHandler.getFormData();
	  orderCreator.createNewOrder(formData, currentUser);
	}

	public void createBill() {
		throw new UnsupportedOperationException();
	}

	public void updateOrderStatus() {
		throw new UnsupportedOperationException();
	}

	public void processOrder() {
		throw new UnsupportedOperationException();
	}

	public void manageOrder() {
		throw new UnsupportedOperationException();
	}

	private void filterOrderList() {
		throw new UnsupportedOperationException();
	}

}
```

DataBaseAPI

```
package com.tasnporstcorp.app;
import java.util.*;
import com.tasnporstcorp.app.orders.*;
import com.tasnporstcorp.app.users.*;

public class DataBaseAPI {

	public DataBaseAPI() {

	}

	public int getNextOrderId() {
		return 1;
	}

	public boolean checkIfOrderExists(Order orderToCheck) {
		return false;
	}

	public boolean postNewOrder(Order order) {
		return true;
	}

	public boolean checkIfAccountExists(ArrayList<String> loginData) {
		return false;
	}

	public boolean postNewUser(User user) {
		return true;
	}

	public void updateVehicleRoute(Vehicle parameter) {
		throw new UnsupportedOperationException();
	}

	public Order getOrder(int orderId) {
		throw new UnsupportedOperationException();
	}

	public ArrayList<Order> getOrdersForUser(LoggedInUser currentUser) {
		throw new UnsupportedOperationException();
	}

	public void updateOrder(Order order) {
		throw new UnsupportedOperationException();
	}

	public ArrayList<Vehicle> getAllVehiclesFromDataBase() {
		throw new UnsupportedOperationException();
	}

	public User getAccount(String login) {
```

```
            throw new UnsupportedOperationException();
      }

}
```

GUIHandler

```
package com.tasnporstcorp.app;

import java.util.Scanner;
import java.util.*;

import com.tasnporstcorp.app.orders.*;

public class GUIHandler {

      public GUIHandler() {

      }

      public void showMessageBox(String message) {
            Scanner in = new Scanner(System.in);
            System.out.println(message);
            System.out.println("Wprowadź ENTER aby kontynuować...");
            in.nextLine();
      }

      public String showDialogBox(String message) {
            Scanner in = new Scanner(System.in);
            System.out.println(message);
            System.out.print("> ");
            return in.nextLine();
      }

      public void showForm() {
            throw new UnsupportedOperationException();
      }

      public ArrayList<String> getFormData() {
            var tmp = new String[]{
                  "paczka", "1", "1", "1",
                  "01-01-2023",
                  "10-01-2023",
                  "Wrocław, ul. Rynek 4", "Wrocław, ul. Rynek 4"
            };
            return new ArrayList<String>(Arrays.asList(tmp));
      }

      public void showOrderData(Order order) {
            throw new UnsupportedOperationException();
      }

      public void showOrderList(ArrayList<Order> orders) {
```

```
            throw new UnsupportedOperationException();
      }

      public void showVehicleList(Vehicle vehicles) {
            throw new UnsupportedOperationException();
      }

      public ArrayList<String> getLoginData() {
            var tmp = new String[]{"login", "jan", "kowalski"};
            return new ArrayList<String>(Arrays.asList(tmp));
      }

}
```

UserCreator

```
package com.tasnporstcorp.app.users;

import com.tasnporstcorp.app.*;
import com.tasnporstcorp.app.users.User.UserRole;
import java.util.*;

public class UserCreator {

      private DataBaseAPI dataBaseApi;
      private GUIHandler guiHandler;

      public UserCreator(GUIHandler guiHandler, DataBaseAPI dataBaseApi) {
            this.dataBaseApi = dataBaseApi;
            this.guiHandler = guiHandler;
      }

      public void registerNewUser(ArrayList<String> loginData) {
            boolean isUserInDatabase =
                  dataBaseApi.checkIfAccountExists(loginData);
            if(isUserInDatabase) {
                  guiHandler.showDialogBox(
                  "Konto o danym loginie już istnieje");
                  return;
            }

            User user = new User(
                  loginData.get(0),
                  loginData.get(1),
                  loginData.get(2)
            );
            UserRole role = UserRole.None;

            while (role == UserRole.None) {
                  String userKey = guiHandler.showDialogBox(
                        "Wprowadź klucz dostępu."
                  );
                  KeyHandler.assignRoleFromKey(user, userKey);
```

```
                role = user.getRole();
                if(role == UserRole.None)
                        guiHandler.showMessageBox(
                        "Podany klucz jest błędny."
                );
        }

        guiHandler.showMessageBox("Dodano nowe konto użytkownika!");
        dataBaseApi.postNewUser(user);
    }

    public LoggedInUser getCurrentUserFromDataBase(String login) {
        return new LoggedInUser(
                login,
                UserRole.Customer,
                "jan",
                "kowalski"
        );
    }

    public User getDriverFromDataBase(String login) {
        throw new UnsupportedOperationException();
    }


    public static class KeyHandler {

        private static final String COORDINATOR_KEY = "123";
        private static final String DRIVER_KEY = "321";

        static void assignRoleFromKey(User user, String key) {
            if(key.equals("")){
                    user.setRole(UserRole.Customer);
            }
            else if(key.equals(COORDINATOR_KEY)){
                    user.setRole(UserRole.Coordinator);
            }
            else if(key.equals(DRIVER_KEY)){
                    user.setRole(UserRole.Driver);
            }
        }

    }

}
```

User

```
package com.tasnporstcorp.app.users;

public class User {

    private String login;
```

```
	private UserRole role = UserRole.None;
	private String firstName;
	private String lastName;

	User(String login, String firstName, String lastName) {
		this.login = login;
		this.firstName = firstName;
		this.lastName = lastName;
	}

	public String getLogin() {
		return this.login;
	}

	public UserRole getRole() {
		return this.role;
	}

	public void setRole(UserRole role) {
		this.role = role;
	}


	public enum UserRole {
		Coordinator,
		Driver,
		Customer,
		None
	}

}
```

LoggedInUser

```
package com.tasnporstcorp.app.users;
import com.tasnporstcorp.app.users.User.*;
import java.util.*;
public class LoggedInUser extends User {

	private ArrayList<String> filtersList;
	private ArrayList<String> sortingCriteria;

	LoggedInUser(String login, UserRole role, String firstName, String
lastName) {
		// TODO - implement LoggedInUser.LoggedInUser
		super(login, firstName, lastName);
		filtersList = new ArrayList<>();
		sortingCriteria = new ArrayList<>();
	}

	public void clearFiltersList() {
		throw new UnsupportedOperationException();
	}
```

```
	public void removeFilter(int positionOnFiltersList) {
		throw new UnsupportedOperationException();
	}

	public void addFilter(String attribute, String condition) {
		throw new UnsupportedOperationException();
	}

	public ArrayList<String> getFiltersList() {
		return this.filtersList;
	}

	public void setSortingCriteria(String attribute, boolean ascending) {
		throw new UnsupportedOperationException();
	}

}
```

Vehicle

```
package com.tasnporstcorp.app.orders;
import java.util.*;
import com.tasnporstcorp.app.users.*;

public class Vehicle {

	private int vehicleId;
	private User assignedDriver;
	private ArrayList<String> assignedRoute;
	private String vehicleName;
	private ArrayList<Order> transportedOrders;
	private int capacityCm3;

	public Vehicle(int vehicleId, User driver, String name, double
capacityCm3) {
		throw new UnsupportedOperationException();
	}

	public void addPointToRoute(int position, String pointAddress) {
		throw new UnsupportedOperationException();
	}

	public void removePointFromRoute(int position, String pointAddress) {
		throw new UnsupportedOperationException();
	}

	public void modifyPointFromRoute(int pointPosition) {
		throw new UnsupportedOperationException();
	}

	public boolean addOrder(Order order) {
		throw new UnsupportedOperationException();
```

```
        }

        public void removeOrder(Order order) {
              throw new UnsupportedOperationException();
        }

}
```

OrderCreator

```
package com.tasnporstcorp.app.orders;
import java.util.*;
import com.tasnporstcorp.app.*;
import com.tasnporstcorp.app.users.User;

public class OrderCreator {

	private GUIHandler guiHandler;
	private DataBaseAPI dataBaseApi;

	public OrderCreator(GUIHandler guiHandler, DataBaseAPI databaseApi) {
		this.guiHandler = guiHandler;
		this.dataBaseApi = databaseApi;
	}

	public void createNewOrder(ArrayList<String> formData, User user) {
		var commodity = new Commodity(
			formData.get(0),
			Double.parseDouble(formData.get(1)),
			Double.parseDouble(formData.get(2)),
			Double.parseDouble(formData.get(3))
		);
		var order = new Order(
			commodity,
			user,
			formData.get(4),
			formData.get(5),
			formData.get(6)
		);
		boolean doesOrderExists =dataBaseApi.checkIfOrderExists(order);
		if(doesOrderExists)
		{
			String userInteraction = guiHandler.showDialogBox(
				"Zamówienie o takich parametrach już istnieje.
				Czy chcesz kontynuować składanie zamówienia?"
			);
			if(userInteraction.contains("NO"))
			{
				return;
			}
		}

		int nextOrderId = dataBaseApi.getNextOrderId();
		order.setId(nextOrderId);

		boolean isSuccess = dataBaseApi.postNewOrder(order);
		if(isSuccess)
		{
			guiHandler.showMessageBox("Dodano nowe zamówienie");
		}
		else
		{
```

```
                    guiHandler.showMessageBox(
                          "Nie udało się dodać zamówienia"
                    );
              }
        }

}
```

Order

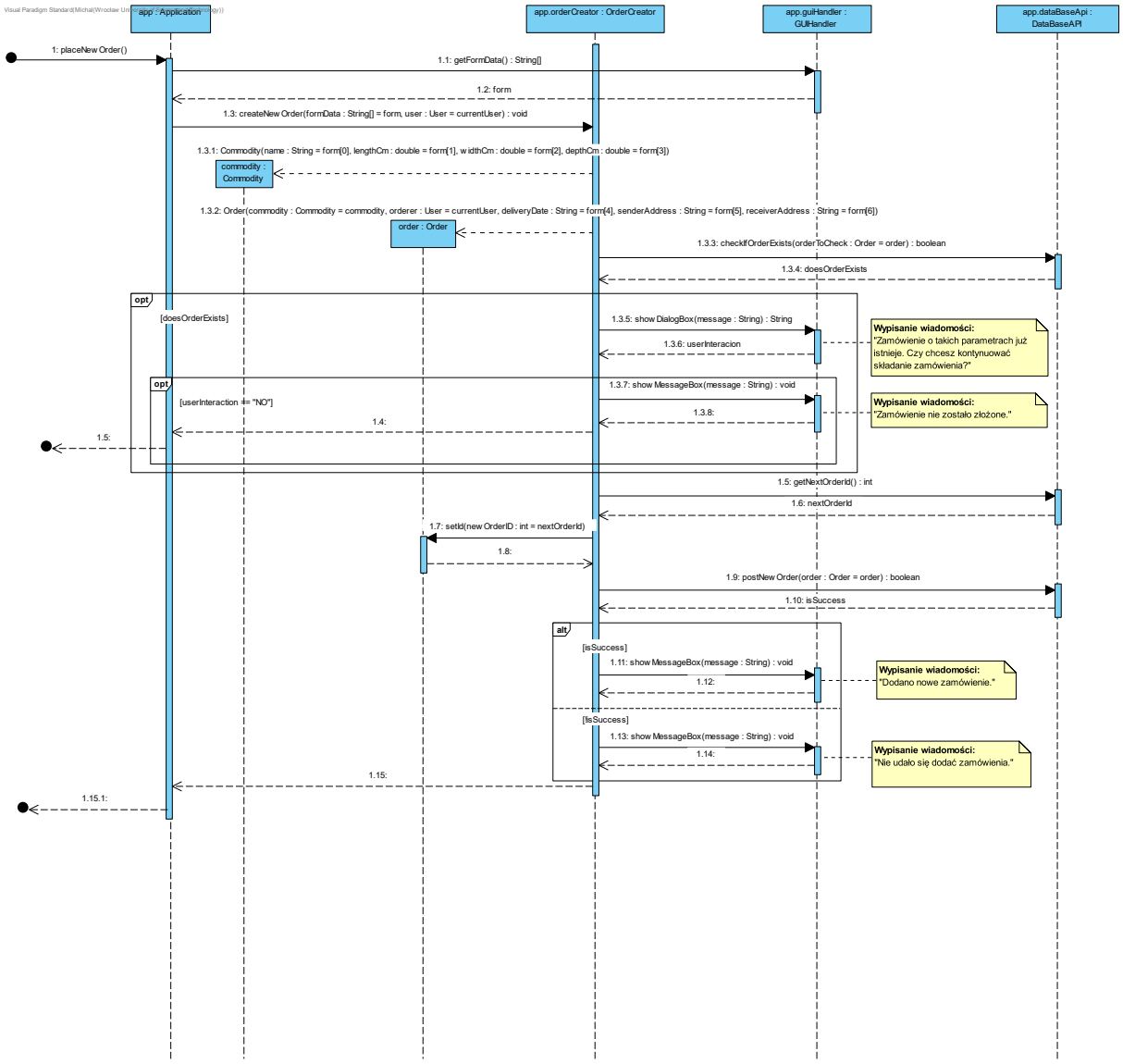
```
package com.tasnporstcorp.app.orders;

import com.tasnporstcorp.app.users.*;

public class Order {

	private int orderId = -1;
	private Commodity transportedCommodity;
	private User orderer;
	private String prefferedDeliveryDate;
	private OrderStatus status = OrderStatus.OnTheTable;
	private String senderAddress;
	private String receiverAddress;
	private Vehicle assignedVehicle = null;
	private String rejectionCause = "";

	public Order(Commodity commodity, User orderer, String deliveryDate,
String senderAddress, String receiverAddress) {
		transportedCommodity = commodity;
		this.orderer = orderer;
		this.prefferedDeliveryDate = deliveryDate;
		this.senderAddress = senderAddress;
		this.receiverAddress = receiverAddress;
	}

	public boolean equals(Object o) {
		throw new UnsupportedOperationException();
	}

	public void setId(int newOrderID) {
		orderId = newOrderID;
	}

	public void assignVehicle(Vehicle assignedVehicle) {
		throw new UnsupportedOperationException();
	}

	public Commodity getTransportedCommodity() {
		return this.transportedCommodity;
	}

	public Vehicle getAssignedVehicle() {
		return this.assignedVehicle;
	}

	public double getPrice() {
		throw new UnsupportedOperationException();
	}

	public OrderStatus getStatus() {
		return this.status;
	}
```

```
	public String getRejectionCause() {
		return this.rejectionCause;
	}

	public String getSenderAddress() {
		return this.senderAddress;
	}

	public String getReceiverAddress() {
		return this.receiverAddress;
	}

	public void acceptOrder() {
		throw new UnsupportedOperationException();
	}

	public void rejectOrder(String rejectionCause) {
		throw new UnsupportedOperationException();
	}

	public void setInProgress() {
		throw new UnsupportedOperationException();
	}

	public void setDelivered() {
		throw new UnsupportedOperationException();
	}


	public enum OrderStatus {
		OnTheTable,
		InProgress,
		Rejected,
		Accepted,
		Delivered
	}

}
```

Commodity

```
package com.tasnporstcorp.app.orders;

public class Commodity {

	public String name;
	public double lengthCm;
	public double widthCm;
	public double depthCm;

	public Commodity(String name, double lengthCm, double widthCm, double
depthCm) {
		this.name = name;
		this.lengthCm = lengthCm;
		this.widthCm = widthCm;
		this.depthCm = depthCm;
	}

	public double getVolumeCm3() {
		throw new UnsupportedOperationException();
	}

}
```

Diagram sekwencji metody „placeOrder” (PU „Składanie zamówienia”):



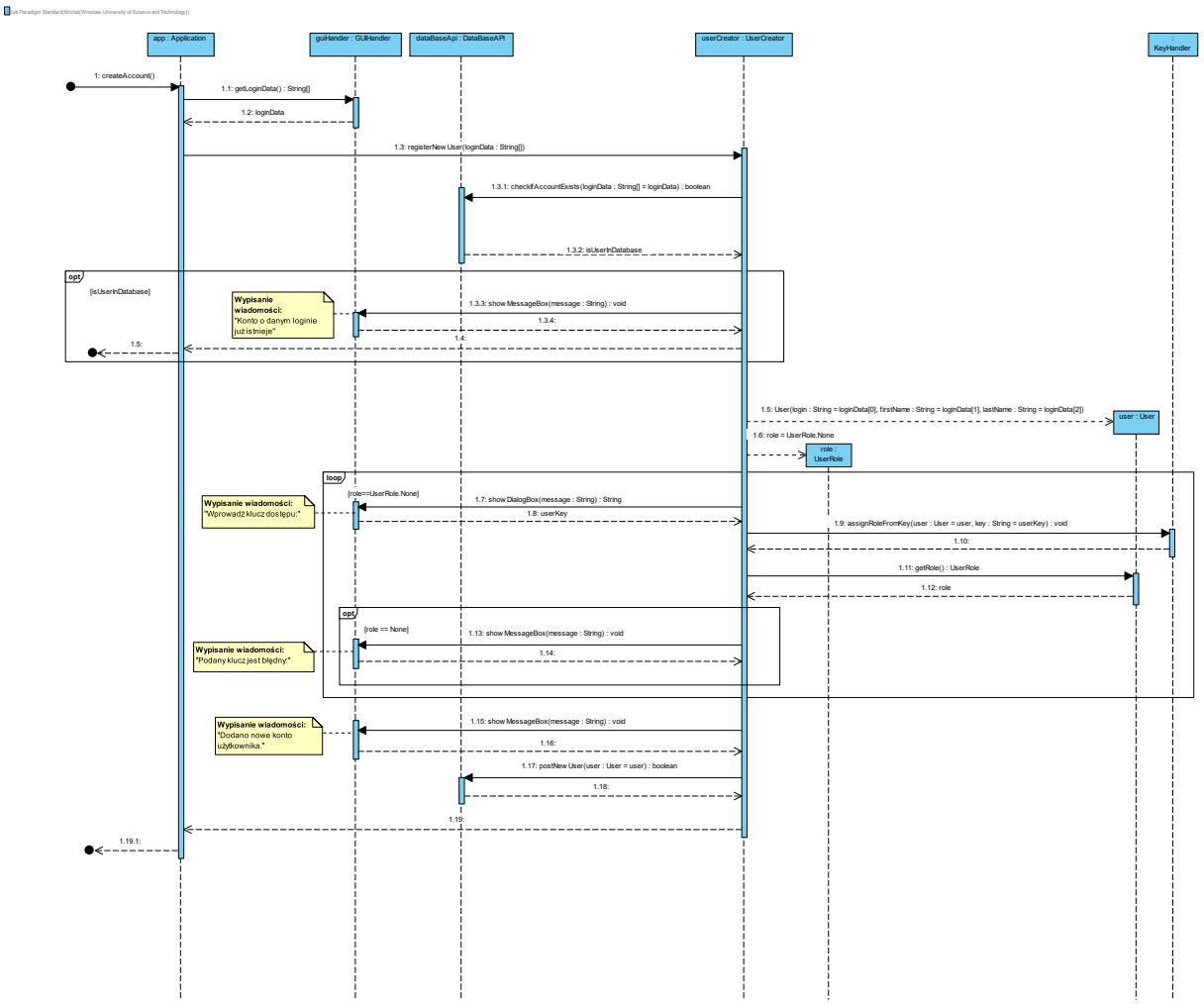
Kod metody `placeNewOrder` z klasy `Application` zamodelowanej przez diagram sekwencji:

```
public void placeNewOrder() {
	String[] formData = guiHandler.getFormData();
	orderCreator.createNewOrder(formData, currentUser);
}
```

Kod metody `createNewOrder` z klasy `OrderCreator` zamodelowanej przez diagram sekwencji:

```
public void createNewOrder(String[] formData, User user) {
        var commodity = new Commodity(
            formData[0],
            Double.parseDouble(formData[1]),
            Double.parseDouble(formData[2]),
            Double.parseDouble(formData[3])
        );

        var order = new Order(
            commodity,
            user,
            formData[4],
            formData[5],
            formData[6]
        );

        boolean doesOrderExists = dataBaseApi.checkIfOrderExists(order);
        if(doesOrderExists)
        {
            String userInteraction = guiHandler.showDialogBox(
                  "Zamówienie o takich parametrach już istnieje. Czy chcesz
                  kontynuować składanie zamówienia?"
            );

            if(userInteraction.contains("NO"))
            {
                return;
            }
        }

        int nextOrderId = dataBaseApi.getNextOrderId();
        order.setId(nextOrderId);

        boolean isSuccess = dataBaseApi.postNewOrder(order);
        if(isSuccess)
        {
            guiHandler.showMessageBox("Dodano nowe zamówienie");
        }
        else
        {
            guiHandler.showMessageBox("Nie udało się dodać zamówienia");
        }
}
```

Diagram sekwencji metody „createAccount” (PU „Założenie konta”):



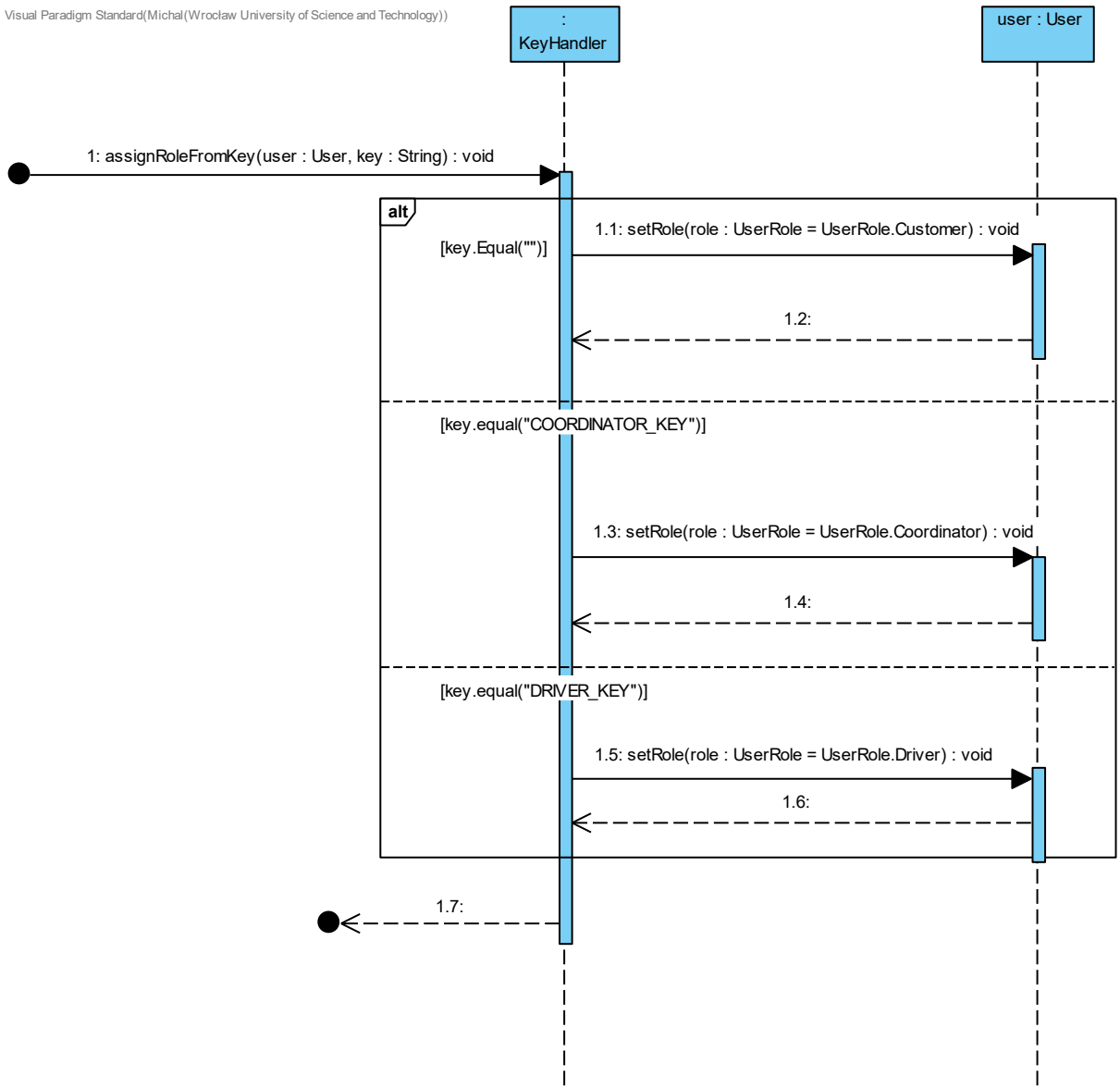
Kod metody `createAccount` z klasy `Application` zamodelowanej przez diagram sekwencji:

```
public void createAccount() {
	String[] loginData = guiHandler.getLoginData();
	userCreator.registerNewUser(loginData);
}
```

Kod metody `registerNewUser` z klasy `UserCreator` zamodelowanej przez diagram sekwencji:

```
public void registerNewUser(String[] loginData) {
	boolean isUserInDatabase =
dataBaseApi.checkIfAccountExists(loginData);
	if(isUserInDatabase) {
		guiHandler.showDialogBox("Konto o danym loginie już istnieje");
		return;
	}

	User user = new User(loginData[0], loginData[1], loginData[2]);
	UserRole role = UserRole.None;

	while (role == UserRole.None) {
		String userKey = guiHandler.showDialogBox("Wprowadź klucz
dostępu.");
		KeyHandler.assignRoleFromKey(user, userKey);
		role = user.getRole();
		if(role == UserRole.None)
			guiHandler.showMessageBox("Podany klucz jest błędny.");
	}

	guiHandler.showMessageBox("Dodano nowe konto użytkownika!");
	dataBaseApi.postNewUser(user);
}
```

Dla czytelności z diagramu modelującego metodę `createAccount` wydzielono do osobnego diagramu sekwencji metodę `assignRoleFromKey` z klasy `KeyHandler`.



Kod metody `assignRoleFromKey` z klasy `KeyHandler` zamodelowanej przez diagram sekwencji:

```
static void assignRoleFromKey(User user, String key) {
	if(key.equals("")){
		user.setRole(UserRole.Customer);
	}
	else if(key.equals(COORDINATOR_KEY)){
		user.setRole(UserRole.Coordinator);
	}
	else if(key.equals(DRIVER_KEY)){
		user.setRole(UserRole.Driver);
	}
}
```