

# Úprava projektu pokladna

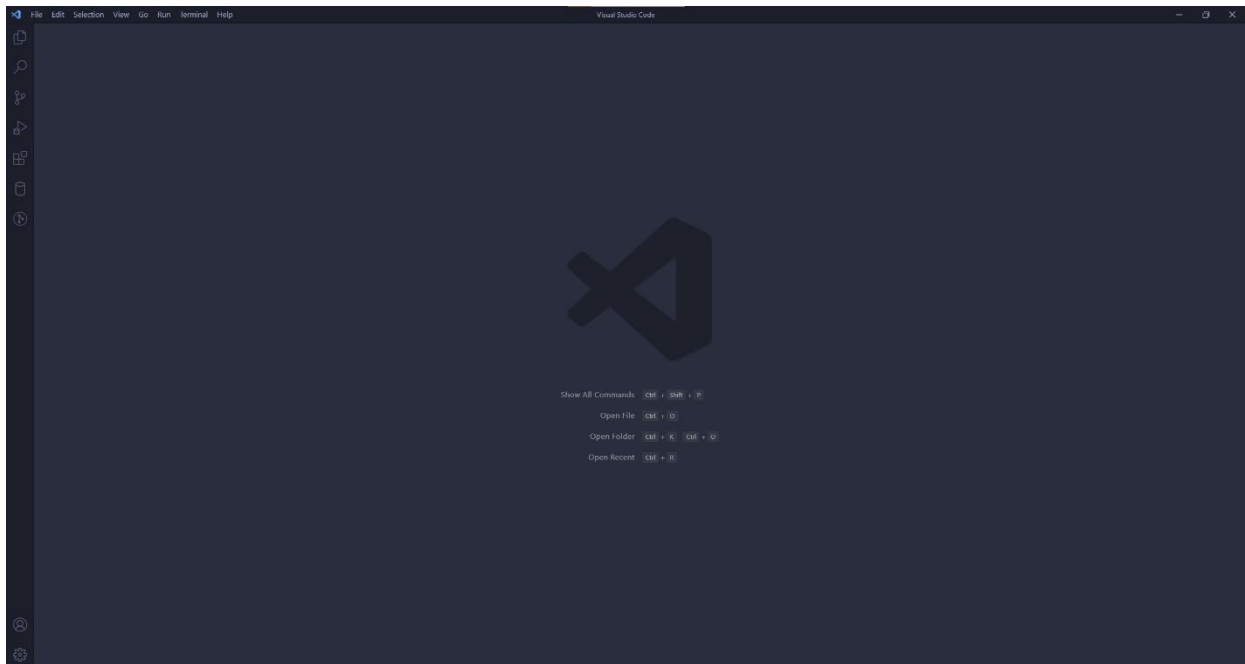
PROGRAMOVÁNÍ A HW 3.E  
DVOŘÁK TOMÁŠ

## Obsah

Vývojové prostředí (IDE) .....	2
Projekt.....	3
Objednávka .....	4
Princip skládání objednávek.....	4
GUI Úvodní.....	4
GUI Hamburgery Frame.....	4
GUI Historie objednávek .....	7
Rozhraní Objednavky.....	8
Třída shared.Objednavka .....	8
Třída server.Objednavky .....	10
Rozhraní shared.Polozky.....	12
Třída shared.Polozka .....	12
Třída server.Polozky .....	14
Rozhraní shared.Pridavky .....	16
Třída shared.Pridavek.....	16
Třída server.Pridavky .....	17
Práce s databází.....	19
Database connection.....	19
Samotná práce s databází v server.Objednavky,Pridavky,Polozky.....	19
Zapisování do databáze.....	20
Výpis z databáze.....	20
Administrace .....	21
Vložení přídavku či položky .....	21
Úprava položky či přídavku.....	23
Uložiště .....	23
Shared.IUloziste.....	23
Třída Shared.Uloziste .....	23
E-R diagram .....	25

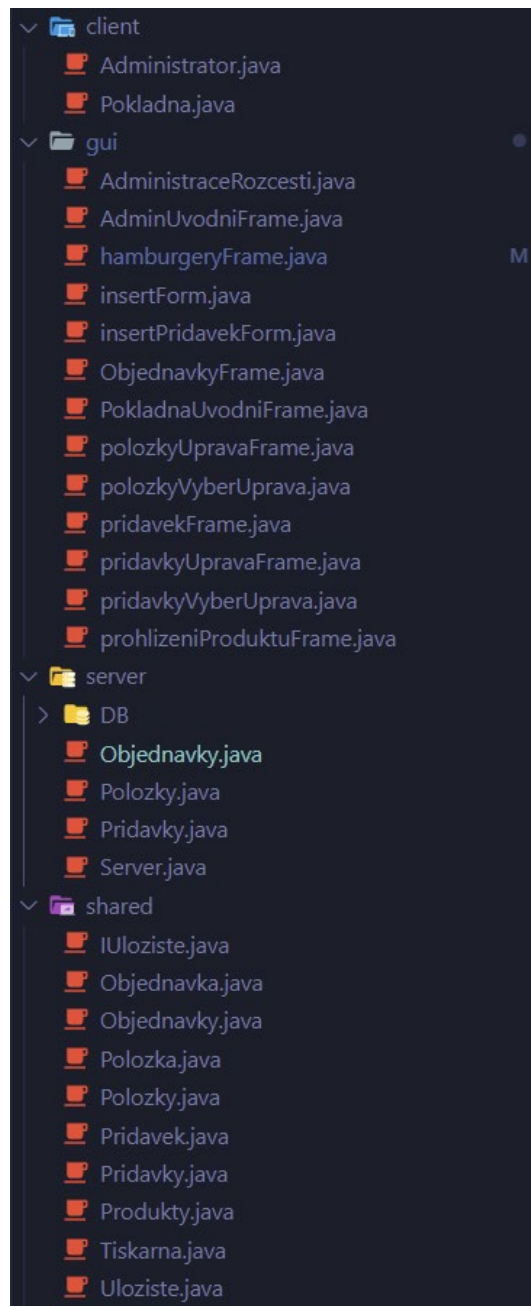
## Vývojové prostředí (IDE)

Aplikace pokladna byla vyvíjena v programu od společnosti Microsoft Visual Studio Code



## Projekt

Soubory projektu jsou rozděleny do složek Client, Server, GUI a Shared podle jejich využití.



# Objednávka

## Princip skládání objednávek

Jednotlivá objednávka je při vytváření statický List, ke kterému se přistupuje pomocí metody getObjednavky a který je naplněný objekty Polozka. Následně se při potvrzení objednávky nastaví hodnoty již samotného objektu objednávky. Nastaví se cena objednávky, čas vytvoření objednávky a list daných Položek, které obsahuje.

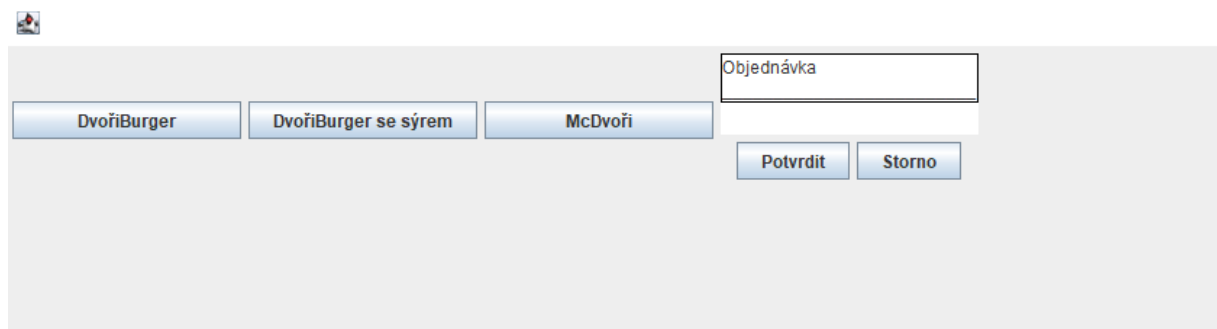
### GUI Úvodní

GUI je psané z větší části ručně ve VS Codu. Objednávky vytváří klient pokladna. Při spuštění klienta se objeví okno PokladnaUvodniFrame, které obsahuje dvě tlačítka. Pro spuštění okna s nabídkou produktů a vytváření objednávky slouží JButton s názvem „Nová objednávka“, který má přidáný ActionListener s lambda výrazem, v kterém se vytvoří nový objekt hamburgeryFrame dále se nastaví se viditelnost PokladnaUvodniFrame na false a nakonec hamburgeryFrame dostanou viditelnost true a to pomocí ...setVisible(true/false).



### GUI Hamburgery Frame

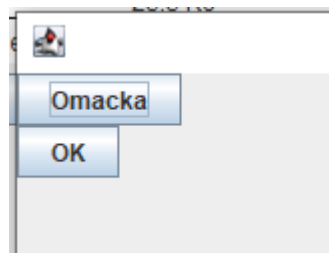
V okně hamburgeryFrame se zobrazí tlačítka aktivních položek, které se vypíší z databáze pomocí foreach cyklu a nastaví se jim ActionListener, JTextPane, ve kterém budou vidět aktuální produkty v objednávce, JTextPane pro celkovou cenu a dva JButtons sloužící k potvrzení nebo stornování objednávky.



### Kliknutí na button produktu

Při kliknutí na produkt se přidá do TextPanelu název produktu a cena, dále se přičte cena k aktuální ceně a zobrazí se v panelu pro celkovou cenu. Také se do listu položek aktuální objednávky přidá daná položka. Následně se zneviditelná aktuální okno a zobrazí se okno

s přídavky, které se opět vypisují z databáze a JButtonu „ok“ který slouží k zneviditelní okna.



Přidání položky do objednávky

```
try {
    Polozky polozky = (Polozky) Naming.lookup("rmi://localhost:12345/polozky");
    for (Polozka p : polozky.getPolozky()) {

        b = new JButton(p.getNazev());

        b.addActionListener((l) -> {
            f = new pridavekFrame();
            f.setVisible(true);
            if (!Objednavka.getObjednavky().add(p)) {
                JOptionPane.showMessageDialog(this, "nepodařilo se přidat do objednávky");
            }

            addText("\n " + p.getNazev() + "\n\t" + p.getCena().toString() + " Kč");

            prictiCenu(Objednavka.getObjednavky().getLast().getCena());
        });

        obalBtn.add(b);
    }
} catch (RemoteException | NotBoundException | MalformedURLException e) {
    e.printStackTrace();
}
```

*Kliknutí na button přidavku*

Při kliknutí na přidavek se k poslednímu objektu položka přidá tento daný přidavek a to tak, že se zavolá metoda objektu položka, která vrátí list přidavků této dané položky a přidá k ní aktuální přidavek. Poté se připiše název a cena přidavku pod poslední položku v JTextPanu aktuální objednávky

Objednávka	
McDvoři	25.5 Kč
Cena	25.5 Kč

Objednávka	
McDvoři	25.5 Kč
	+Omacka 5.4 Kč
Cena	30.9 Kč

## Přidání přídavku

```
try {
    Pridavky pridavky = (Pridavky) Naming.lookup("rmi://localhost:12345/pridavky");
    for (Pridavek p : pridavky.getPridavky()) {

        b = new JButton(p.getNazev());
        b.addActionListener((l) -> {
            Objednavka.getObjednavky().getLast().getPridavky().add(p);

            hamburgeryFrame.addText("\t+" + p.getNazev() + " " + p.getCena() + " Kč");
            hamburgeryFrame.prictiCenu(p.getCena());
        });

        pane.add(b);
    }
} catch (RemoteException | NotBoundException | MalformedURLException e) {
    e.printStackTrace();
}
```

### *Kliknutí na tlačítko storno*

Při kliknutí na toto tlačítko dojde k vymazání aktuální objednávky. Vyprázdní se list položek aktuální objednávky, dále se cena nastaví na výchozích 0.0 a smaže se obsah JTextPanu objednávky a celkové ceny.

### *Kliknutí na tlačítko potvrdit*

Při kliknutí na toto tlačítko se děje nejvíce věcí. Nejprve se zkontroluje jestli je v listu položek aspoň nějaká položka. Jestliže je list prázdný, tak vyskočí JOptionPane se zprávou, že v objednávce není žádná položka. Pakliže je tam alespoň jedna položka, tak se před Naming.lookup odkáže na třídu Objednavka, která je na serveru pod názvem objednávky. Dále se zde uloží čas při stisknutí tlačítka zformátovaný do databázového data a vytvoří se objekt Objednavka, kterému s enásledně nastaví ony hodnoty zmíněné výše. Poté se vytvoří ještě objekt tiskárna, který zavolá metodu Tiskni s parametrem objektu Objednavka. Dále s opět přes Naming.lookup odkáže na třídu Uloziste, která obstarává ukládání objednávek na serveru a jejich mazání. Tento objekt zavolá svou metodu zapisDoUloziste(objednavka), pokud tato metoda vrátí false, tak pomocí throw vyhodí nová výjimka, která říká, že se nepodařilo zapsat objednávku. Nakonec se přes objekt, který komunikuje se serverem se zavolá metoda pro zapsání do databáze. Pokud se zapsání podařilo, vyčistí se list položek aktuální objednávky, cena se nastaví na 0.0, zneviditelní se aktuální Frame a zviditelní se úvodní

## Potvrdit ActionListener

```
potvrdit.addActionListener((e) -> {  
  
    try {  
        if (!Objednavka.getObjednavky().isEmpty()) {  
            Objednavky objednavky = (Objednavky) Naming.lookup("rmi://localhost:12345/objednavky");  
            java.util.Date dt = new java.util.Date();  
  
            java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
  
            String currentTime = sdf.format(dt);  
            LinkedList<Polozka> o = Objednavka.getObjednavky();  
            objednavka = new Objednavka().setCasObjednavky(currentTime).setCena(cena).setPolozky(o);  
            tiskarna = Tiskarna.getITiskarna();  
            tiskarna.Tiskni(objednavka);  
  
            try {  
                IUloziste uloziste = (IUloziste) Naming.lookup("rmi://localhost:12345/uloziste");  
                if (!uloziste.zapisDoUloziste(objednavka))  
                    throw new Exception("Nepodařilo se zapsat na server objednavku");  
            } catch (Exception ex) {  
                ex.printStackTrace();  
            }  
  
            if(objednavky.writeObjednavka(objednavka)) {  
                cena = 0.0;  
                o = null;  
                Objednavka.getObjednavky().clear();  
                this.setVisible(false);  
                new PokladnaUvodniFrame().setVisible(true);  
            }  
  
        } else {  
            JOptionPane.showMessageDialog(this, "Objednávka neobsahuje žádnou položku");  
        }  
    } catch (RemoteException | NotBoundException | MalformedURLException ex) {  
        ex.printStackTrace();  
    }  
});
```

## GUI Historie objednávek

Do historie objednávek se dostává kliknutím na tlačítko „Historie objednávek“ v oknu PokladnaUvodniFrame. Po kliknutí na toto tlačítko se zobrazí ObjednavkyFrame, kde je JTabbed pane, kde v každé záložce je objednávka od nejstarší po nejnovější. Zde se pomocí Naming.lookup odkazující na Třidu objednavka na serveru. Dále se pomocí foreach cyklu, ve které se prochází načtené objednávky z databáze a dalších vnořených foreach cyklů pro výpis položek a přídavek sestavuje objednávka pro prohlížení a plus se přidá JButton, který při kliknutí vytiskne danou objednávku. Objednávka se sestavuje do JTextArei pomocí append a zavolání metod pro získání dat z objednávky(cena, položky, název, čas) a dále se zde upravuje získaný databázový formát data na český. To se dělá pomocí objektů SimpleDateFormat, Date a DateFormat, kde se SDF nastaví databázový formát do objektu Date se pomocí SDF.parse(formát) předá datum z databáze, vytvoří se DateFormat s patternem požadovaného formátu data a pomocí funkce .format(Date) se uloží do řetězce zformátované datum.



```

private void initComponents() {
    tabbedPane = new JTabbedPane();

    try {
        Objednavky objednavky = (Objednavky) Naming.lookup("rmi://localhost:12345/objednavky");
        int cislo = 1;
        for (Objednavka objednavka : objednavky.getObjednavky()) {

            JPanel obal = new JPanel();
            BoxLayout b = new BoxLayout(obal, BoxLayout.Y_AXIS);
            obal.setLayout(b);
            JTextArea obj = new JTextArea();
            obj.setEditable(false);
            SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            Date date = format.parse(objednavka.getCasObjednavky());
            DateFormat df = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
            String result = df.format(date);
            obj.setText("Objednávka z: " + result);
            obj.append("\n");
            obj.append("_____");
            obj.append("\n");

            for (Polozka polozka : objednavka.getPolozky()) {
                obj.append(polozka.getNazev() + "\t" + polozka.getCena() + " Kč");
                obj.append("\n");

                for (Pridavek pridavek : polozka.getPridavky()) {
                    obj.append("\t" + "+" + pridavek.getNazev() + "\t" + pridavek.getCena() + " Kč");
                    obj.append("\n");
                }
            }

            obj.append("\n");
            obj.append("_____");
            obj.append("\n");
            obj.append("Celková cena:\t" + objednavka.getCena() + " Kč");
            obal.add(obj);
            JButton tiskniButton = new JButton("Tiskni");
            tiskniButton.addActionListener((l) -> {
                Tiskarna tiskarna = Tiskarna.getITiskarna();
                tiskarna.Tiskni(objednavka);
            });
            JButton zistiButton = new JButton("Zisti");
        }
    } catch (RemoteException e) {
        e.printStackTrace();
    }
}

```

## Rozhraní Objednavky

Toto rozhraní slouží jako rozhraní komunikace objednávek se serverem. Obsahuje 2 metody, `writeObjednavka(Objednavka objednavka)` a `List<Objednavka> getObjednavky()`, obě metody vyhazují výjimku `RemoteException`

```

package shared;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

You, a day ago | 1 author (You)
public interface Objednavky extends Remote {
    List<Objednavka> getObjednavky() throws RemoteException;
    boolean writeObjednavka(Objednavka objednavka) throws RemoteException;
}

```

## Třída shared.Objednavka

Tato třída implementuje rozhraní `Serializable` a obsahuje několik metod. Obsahuje metodu pro získání a nastavení ceny, získání a nastavení ID objednávky, získání času objednávky a nastavení času, kdy byla objednávka vytvořena k nastavení lokálního listu položek a k vrácení listu položek

```

> shared > Objednavka.java > Objednavka > setObjednavky(LinkedList<Polozka>)
public class Objednavka implements Serializable {
    /**
     */
    private static final long serialVersionUID = -4666760960562716166L;
    private int id;
    private Double cena;
    private String casObjednavky;
    private static LinkedList<Polozka> objednavka = new LinkedList<>();
    private LinkedList<Polozka> polozky = new LinkedList<>();

    public Double getCena() {
        return cena;
    }

    public LinkedList<Polozka> getPolozky() {
        return polozky;
    }

    public Objednavka setPolozky(LinkedList<Polozka> polozky) {
        this.polozky = polozky;
        return this;
    }

    public int getId() {
        return id;
    }

    public Objednavka setId(int id) {
        this.id = id;
        return this;
    }

    public String getCasObjednavky() {
        return casObjednavky;
    }

    public Objednavka setCasObjednavky(String casObjednavky) {
        this.casObjednavky = casObjednavky;
        return this;
    }

    public static LinkedList<Polozka> getObjednavky() {
        return objednavka;
    }

    public Objednavka setObjednavky(LinkedList<Polozka> objednavka) {
        objednavka.objednavka = objednavka;
        return this;
    }

    public Objednavka setCena(Double cena) {
        this.cena = cena;
        return this;
    }
}

```

## Třída server.Objednavky

Tato třída implementuje rozhraní shared.Objednavky. Jsou zde metody pro zápis objednávek do databáze a získání objednávek z databáze. Více o mém návržení práce s databází zde.

```
}

@Override
public List<Objednavka> getObjednavky() throws RemoteException {
    List<Objednavka> objednavky = new ArrayList<>();

    try (Connection conn = Database.get().getConnection();
        Statement objednavkyStmt = conn.createStatement();

        ResultSet objednavkyRs = objednavkyStmt.executeQuery(
            "SELECT objednavky.ID, objednavky.celkovaCena, objednavky.casObjednavky FROM objednavky ORDER BY objednavky.ID");

        PreparedStatement polozkyStmt = conn.prepareStatement(
            "SELECT DISTINCT polozky_v_objednavce.Polozka_ID, polozky.nazev, polozky.cena, polozky.druh, polozky_v_objednavce.cisloPolozkyVobjednavce FROM polozky_v_objednavce JOIN polozky ON polozky.ID = polozky_v_objednavce.Polozka_ID");

        PreparedStatement pridavkyStmt = conn.prepareStatement(
            "SELECT polozky_v_objednavce.Pridavek_ID, polozky_v_objednavce.cisloPolozkyVobjednavce, pridavky.nazev, pridavky.cena FROM polozky_v_objednavce JOIN pridavky ON pridavky.ID = polozky_v_objednavce.Pridavek_ID");

        while (objednavkyRs.next()) {
            Objednavka objednavka = new Objednavka().setCasObjednavky(objednavkyRs.getString("casObjednavky"))

```

```

            ;

            while (objednavkyRs.next()) {

                Objednavka objednavka = new Objednavka().setCasObjednavky(objednavkyRs.getString("casObjednavky"))
                    .setCena(objednavkyRs.getDouble("celkovaCena")).setId(objednavkyRs.getInt("ID"));

                polozkyStmt.setInt(1, objednavka.getId());

                try (ResultSet polozkaRs = polozkyStmt.executeQuery()) {

                    while (polozkaRs.next()) {

                        Polozka polozka = new Polozka().setId(polozkaRs.getInt("Polozka_ID"))
                            .setNazev(polozkaRs.getString("nazev")).setCena(polozkaRs.getDouble("cena"))
                            .setDruh(polozkaRs.getString("druh"));

                        objednavka.getPolozky().add(polozka);

                        int cisloPolozkyVobjednavce = polozkaRs.getInt("cisloPolozkyVobjednavce");

                        pridavkyStmt.setInt(1, cisloPolozkyVobjednavce);
                        pridavkyStmt.setInt(2, objednavka.getId());
                        pridavkyStmt.setInt(3, polozka.getId());

                        try (ResultSet pridavekRs = pridavkyStmt.executeQuery()) {

                            while (pridavekRs.next()) {

                                Pridavek pridavek = new Pridavek().setId(pridavekRs.getInt("Pridavek_ID"))
                                    .setNazev(pridavekRs.getString("nazev")).setCena(pridavekRs.getDouble("cena"));

                                polozka.getPridavky().add(pridavek);

                            }
                        } catch (Exception e) {
                            e.printStackTrace();

                            continue;
                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace();

                    continue;
                }
            }
        }
    }
}

You, 3 days ago • v0.6,dodělán výpis objednavek

```

```

@Override
public boolean writeObjednavka(Objednavka objednavka) throws RemoteException {
    try (Connection conn = Database.get().getConnection()) {
        conn.setAutoCommit(false);

        try (PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO objednavky (celkovaCena, casObjednavky) VALUES (?,?)",
            PreparedStatement.RETURN_GENERATED_KEYS)) {
            stmt.setDouble(1, objednavka.getCena());

            stmt.setString(2, objednavka.getCasObjednavky());

            if (stmt.executeUpdate() != 1) {
                throw new Exception("Nepodaril se zapis objednavky");
            }

            ResultSet rs = stmt.getGeneratedKeys();
            if (rs.next()) {
                objednavka.setId(rs.getInt(1));
            }
            rs.close();
        } catch (Exception e) {
            e.printStackTrace();
            conn.rollback();
            throw e;
        }

        try (PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO polozkamapridavky(Polozky_ID, Pridavek_ID) VALUES(?,?)",
            PreparedStatement.RETURN_GENERATED_KEYS)) {

            for (Polozka polozka : objednavka.getPolozky()) {

                if (polozka.getPridavky().isEmpty() == false) {
                    for (Pridavek pridavek : polozka.getPridavky()) {

                        stmt.setInt(1, polozka.getId());
                        stmt.setInt(2, pridavek.getId());
                        stmt.executeUpdate();

                    }
                } else {
                    stmt.setInt(1, polozka.getId());
                    stmt.setNull(2, 4);
                    stmt.executeUpdate();
                }
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

## Rozhraní shared.Polozky

Rozhraní obsahuje 4 metody:

1. `boolean upravPolozka(Polozka polozka)` -> v server.Objednavky zajišťuje úpravu položky v DB
2. `List<Polozka> getPolozky()` -> výpis položek z databáze
3. `List<Polozka> getPolozkyAdmin()` -> výpis i neaktivních položek
4. `boolean writePolozka(Polozka polozka)` -> zapsání položky do Databáze

Všechny metody pomocí throws vyhazují RemoteException

## Třída shared.Polozka

Tato třída dědí od abstraktní třídy shared.Produkty, která implementuje rozhraní Serializable. Obsahuje metody pro získání a nastavení hodnot objektu položky (ID, cena, nazev, isActive a listu přidavků).

```
You, a day ago | 1 author (You)
public class Polozka extends Produkty {
    /**
     *
     */
    private static final long serialVersionUID = 4457999599152509060L;
    private int id;
    private String nazev;
    private Double cena;
    private String druh;
    private List<Pridavek> pridavky = new ArrayList<>();
    private boolean isActive;

    public Double getCena() {
        return cena;
    }

    public boolean isActive() {
        return isActive;
    }

    public Polozka setActive(boolean isActive) {
        this.isActive = isActive;
        return this;
    }

    public List<Pridavek> getPridavky() {
        return pridavky;
    }

    public Polozka setPridavky(List<Pridavek> pridavky) {
        this.pridavky = pridavky;
    }
}
```

```

    public String getDruh() {
        return druh;
    }

    public Polozka setDruh(String druh) {
        this.druh = druh;
        return this;
    }

    public Polozka setCena(Double cena) {
        this.cena = cena;
        return this;
    }

    public String getNazev() {
        return nazev;
    }

    public Polozka setNazev(String nazev) {
        this.nazev = nazev;
        return this;
    }

    public int getId() {
        return id;
    }

    public Polozka setId(int id) {
        this.id = id;
        return this;
    }
}

```

## Třída server.Polozky

Tato třída implementuje rozhraní `shared.Polozky`. Jsou zde metody pro zápis položky do databáze a získání položek z databáze, úprava položky. Více o mém návrhu práce s databází [zde](#).

```
28
29     @Override
30     public boolean writePolozka(Polozka polozka) throws RemoteException {
31         try (Connection conn = Database.get().getConnection()) {
32             conn.setAutoCommit(false);
33
34             try (java.sql.PreparedStatement stmt = conn.prepareStatement(
35                 "INSERT INTO polozky (nazev, cena, druh) VALUES (?, ?, ?)",
36                 PreparedStatement.RETURN_GENERATED_KEYS)) {
37                 stmt.setString(1, polozka.getNazev());
38                 stmt.setDouble(2, polozka.getCena());
39                 stmt.setObject(3, polozka.getDruh());
40
41                 if (stmt.executeUpdate() != 1) {
42                     throw new Exception("Nepodaril se zapis polozky");
43                 }
44
45                 ResultSet rs = stmt.getGeneratedKeys();
46
47                 if (rs.next()) {
48                     polozka.setId(rs.getInt(1));
49                 }
50                 rs.close();
51             } catch (Exception e) {
52                 e.printStackTrace();
53                 conn.rollback();
54                 throw e;
55             }
56             conn.commit();
57             return true;
58         } catch (Exception e) {
59             System.out.println(e.getMessage());
60             e.printStackTrace();
61             return false;
62         }
63     }
64
```

```
}

@Override
public List<Polozka> getPolozky() throws RemoteException {
    List<Polozka> polozky = new ArrayList<>();
    try (Connection conn = Database.get().getConnection();
        Statement polozkyStmt = conn.createStatement();
        ResultSet polozkyRs = polozkyStmt.executeQuery(
            "SELECT polozky.ID, polozky.nazev, polozky.cena, polozky.druh, polozky.isActive FROM polozky WHERE polozky.isActive = 1")) {
        ;

        while (polozkyRs.next()) {
            Polozka polozka = new Polozka().setId(polozkyRs.getInt("ID")).setNazev(polozkyRs.getString("nazev"))
                .setCena(polozkyRs.getDouble("cena")).setDruh(polozkyRs.getString("druh"))
                .setActive(polozkyRs.getBoolean("isActive"));
            polozky.add(polozka);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return polozky;
}
```

```

@Override
public List<Polozka> getPolozkyAdmin() throws RemoteException {
    List<Polozka> polozky = new ArrayList<>();
    try (Connection conn = Database.get().getConnection();
        Statement polozkyStmt = conn.createStatement();
        ResultSet polozkyRs = polozkyStmt.executeQuery(
            "SELECT polozky.ID, polozky.nazev, polozky.cena, polozky.druh, polozky.isActive FROM polozky")) {
        while (polozkyRs.next()) {
            Polozka polozka = new Polozka().setId(polozkyRs.getInt("ID")).setNazev(polozkyRs.getString("nazev"))
                .setCena(polozkyRs.getDouble("cena")).setDruh(polozkyRs.getString("druh"))
                .setActive(polozkyRs.getBoolean("isActive"));
            polozky.add(polozka);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return polozky;
}

```

```

@Override
public boolean upravPolozka(Polozka polozka) throws RemoteException {
    try (Connection conn = Database.get().getConnection()) {
        conn.setAutoCommit(false);

        try (PreparedStatement stmt = conn.prepareStatement(
            "UPDATE polozky SET polozky.isActive = ?, polozky.nazev = ?, polozky.cena = ?, polozky.druh = ? WHERE polozky.ID = ?")) {
            int pom;
            if (polozka.isActive())
                pom = 1;
            else
                pom = 0;
            stmt.setInt(1, pom);
            stmt.setString(2, polozka.getNazev());
            stmt.setDouble(3, polozka.getCena());
            stmt.setString(4, polozka.getDruh());
            stmt.setInt(5, polozka.getId());

            if (stmt.executeUpdate() != 1) {
                throw new Exception("Nepodařilo se upravit polozku");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            conn.rollback();
            System.out.println(e.getMessage());
            throw e;
        }
        conn.commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println(e.getMessage());
        return false;
    }
}

```



## Rozhraní shared.Pridavky

Obsahuje 4 přidavky a vypadá podobně jako rozhraní shared.Polozky

1. `List<Pridavek> getPridavky()` -> vrací přidavky z Databáze
2. `List<Pridavek> getPridavkyAdministrace()` -> vrací aktivní přidavky
3. `boolean writePridavek(Pridavek pridavek)` -> zapisuje přidavky do databáze
4. `boolean upravPridavek(Pridavek pridavek)` -> upravuje přidavky

Všechny metody pomocí throws vyhazují RemoteException

## Třída shared.Pridavek

Tato třída dědí od abstraktní třídy shared.Produkty, která implementuje rozhraní Serializable. Obsahuje metody pro získání a nastavení hodnot objektu přidavku (ID, cena, nazev, isActive).

```
private static final long serialVersionUID = -2477700579747907176L;
private int id;
private String nazev;
private Double cena;
private boolean isActive;

public Double getCena() {
    return cena;
}

public boolean isActive() {
    return isActive;
}

public Pridavek setActive(boolean isActive) {
    this.isActive = isActive;
    return this;
}

public Pridavek setCena(Double cena) {
    this.cena = cena;
    return this;
}

public String getNazev() {
    return nazev;
}

public Pridavek setNazev(String nazev) {
    this.nazev = nazev;
    return this;
}

public int getId() {
    return id;
}

public Pridavek setId(int id) {
    this.id = id;
    return this;
}
```

## Třída server.Pridavky

Tato třída implementuje rozhraní `shared.Pridavky`. Jsou zde metody pro zápis přídávku do databáze, získání přídávků z databáze a úpravu přídávku. Více o mém návržení práce s databází [zde](#).

```
@Override
public List<Pridavek> getPridavky() throws RemoteException {
    List<Pridavek> pridavky = new ArrayList<>();
    try (Connection conn = Database.get().getConnection()) {
        Statement pridavkyStatement = conn.createStatement();
        ResultSet pridavkyRs = pridavkyStatement.executeQuery(
            "SELECT pridavky.ID, pridavky.nazev, pridavky.cena, pridavky.isActive FROM pridavky WHERE pridavky.isActive = 1");
        while (pridavkyRs.next()) {
            Pridavek pridavek = new Pridavek().setId(pridavkyRs.getInt("ID"))
                .setNazev(pridavkyRs.getString("nazev")).setCena(pridavkyRs.getDouble("cena"))
                .setActive(pridavkyRs.getBoolean("isActive"));
            pridavky.add(pridavek);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return pridavky;
}
```

```
@Override
public boolean writePridavek(Pridavek pridavek) throws RemoteException {
    try (Connection conn = Database.get().getConnection()) {
        conn.setAutoCommit(false);

        try (PreparedStatement stmt = conn.prepareStatement("INSERT INTO pridavky (nazev, cena) VALUES (?,?)",
            PreparedStatement.RETURN_GENERATED_KEYS)) {
            stmt.setString(1, pridavek.getNazev());
            stmt.setDouble(2, pridavek.getCena());

            if (stmt.executeUpdate() != 1) {
                throw new Exception("Nepodaril se zapis pridavku");
            }

            ResultSet rs = stmt.getGeneratedKeys();

            if (rs.next()) {
                pridavek.setId(rs.getInt(1));
            }
            rs.close();
        } catch (Exception e) {
            e.printStackTrace();
            conn.rollback();
            throw e;
        }
        conn.commit();
        return true;
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
        return false;
    }
}
```

```

@Override
public List<Pridavek> getPridavkyAdministrace() throws RemoteException {
    List<Pridavek> pridavky = new ArrayList<>();
    try (Connection conn = Database.get().getConnection()) {
        Statement pridavkyStatement = conn.createStatement();
        ResultSet pridavkyRs = pridavkyStatement
            .executeQuery("SELECT pridavky.ID, pridavky.nazev, pridavky.cena, pridavky.isActive FROM pridavky");
        while (pridavkyRs.next()) {
            Pridavek pridavek = new Pridavek().setId(pridavkyRs.getInt("ID"))
                .setNazev(pridavkyRs.getString("nazev")).setCena(pridavkyRs.getDouble("cena"))
                .setActive(pridavkyRs.getBoolean("isActive"));
            pridavky.add(pridavek);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return pridavky;
}

```

```

}

@Override
public boolean upravPridavek(Pridavek pridavek) throws RemoteException {
    try (Connection conn = Database.get().getConnection()) {
        conn.setAutoCommit(false);

        try (PreparedStatement stmt = conn.prepareStatement(
            "UPDATE pridavky SET pridavky.isActive = ?, pridavky.nazev = ?, pridavky.cena = ? WHERE pridavky.ID = ?")) {
            int pom;
            if (pridavek.isActive())
                pom = 1;
            else
                pom = 0;
            stmt.setInt(1, pom);
            stmt.setString(2, pridavek.getNazev());
            stmt.setDouble(3, pridavek.getCena());

            stmt.setInt(4, pridavek.getId());

            if (stmt.executeUpdate() != 1) {
                throw new Exception("Nepodařilo se upravit položku");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            conn.rollback();
            System.out.println(e.getMessage());
            throw e;
        }
        conn.commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println(e.getMessage());
        return false;
    }
}

```

## Práce s databází

### Database connection

Třída Database je ve složce server. Používá se zde Singleton, aby bylo vytvořena vždy jen jedna instance databáze. V konstruktoru se inicializuje ovladač komunikace s databází mysql.jdbc.Driver. A jsou zde metody pro získání spojení s databází, navazuje se pomocí DriverManager.getConnection(url, user, password).

```
import java.sql.DriverManager;

You, 3 days ago | 1 author (You)
public class Database {
    private static Database instance;

    public static Database get() throws ClassNotFoundException, SQLException {
        if(instance == null){
            instance = new Database();
        }
        return instance;
    }

    private String cs = "jdbc:mysql://localhost:3306/pokladna", user = "root", pass = "";

    private Database() throws ClassNotFoundException, SQLException {
        Class.forName("com.mysql.jdbc.Driver");
    }

    private final String typDB = "mysql";
    private final String serverName = "localhost";
    private final int port = 3306;
    private final String defaultDB = "pokladna";

    public Connection getConnection() throws SQLException {
        return getConnection(typDB, serverName, port, defaultDB, user, pass);
    }

    public Connection getConnection(String typDB, String serverName, int port, String defaultDB, String user, String pass) throws SQLException {
        return DriverManager.getConnection(cs, user, pass);
    }
}

You, 2 weeks ago • pokladnaV0.2
```

### Samotná práce s databází v server.Objednavky,Pridavky,Polozky

V metodách pro získání položek, objednávek nebo Přídavků si vytvořím list s daným objektem, dále pomocí try resources si získám spojení s databází, které uložím do objektu třídy Connection: dále si vytvořím statement, který slouží k provádění sql příkazů. Dále si vytvořím objekt Třídy ResultSet, který slouží k uložení výsledků vytažených z databáze pomocí sql dotazu. A v cyklu While projíždím výsledky a z nich vytvářím dané potřebné objekty, jednotlivé hodnoty z databáze se získají metodou get“datový typ“(název sloupce v sql) a následně tento objekt přidám do listu.

Při vkládání do databáze používám transakce(commit). Nejdříve si vytvořím stejně spojení. Nastavím AutoCommit na false a v jednotlivých try reasourcích si vytvořím preparedStatementy s sql dotazem. Pomocí metod set“datový typ“(index, hodnota) nastavím parametry hodnot z sql dotazu. Následně provedu pomocí funkce executeUpdate dané sql příkazy a funkcí commit() potvrdím transakci, pokud se vyskytně nějaká výjimka, transakce se zamkne a zruší.

## Zapisování do databáze

```
@Override
public boolean writePolozka(Polozka polozka) throws RemoteException {
    try (Connection conn = Database.get().getConnection()) {
        conn.setAutoCommit(false);

        try (java.sql.PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO polozky (nazev, cena, druh) VALUES (?, ?, ?)",
            PreparedStatement.RETURN_GENERATED_KEYS)) {
            stmt.setString(1, polozka.getNazev());
            stmt.setDouble(2, polozka.getCena());
            stmt.setObject(3, polozka.getDruh());

            if (stmt.executeUpdate() != 1) {
                throw new Exception("Nepodaril se zapis polozky");
            }

            ResultSet rs = stmt.getGeneratedKeys();

            if (rs.next()) {
                polozka.setId(rs.getInt(1));
            }
            rs.close();
        } catch (Exception e) {
            e.printStackTrace();
            conn.rollback();
            throw e;
        }
        conn.commit();
        return true;
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
        return false;
    }
}
```

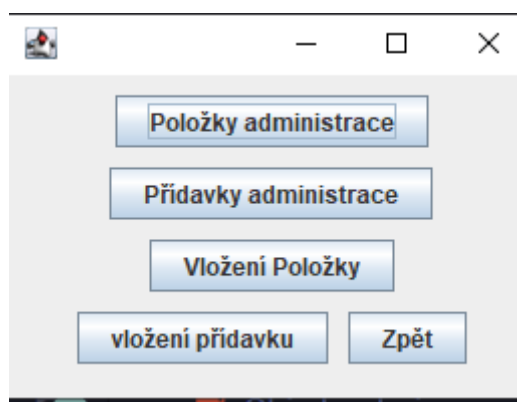
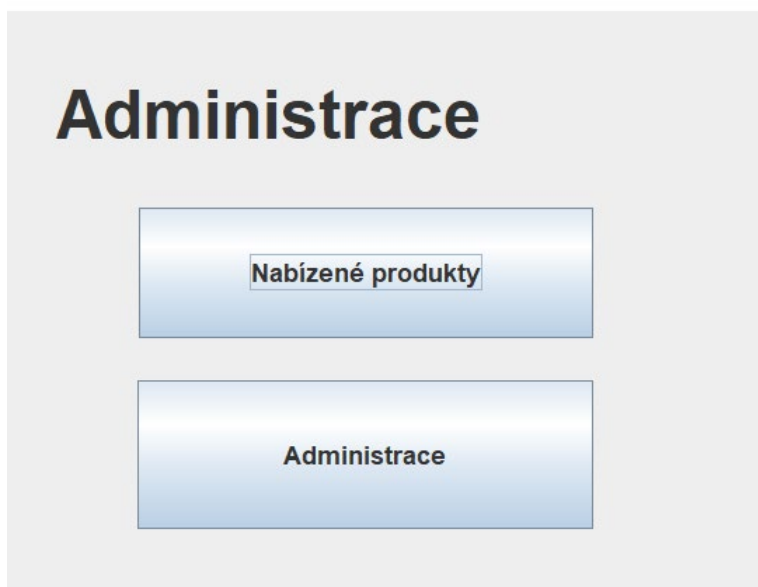
## Výpis z databáze

```
@Override
public List<Polozka> getPolozky() throws RemoteException {
    List<Polozka> polozky = new ArrayList<>();
    try (Connection conn = Database.get().getConnection();
        Statement polozkyStmt = conn.createStatement();
        ResultSet polozkyRs = polozkyStmt.executeQuery(
            "SELECT polozky.ID, polozky.nazev, polozky.cena, polozky.druh, polozky.isActive FROM polozky WHERE polozky.isActive = 1")) {
        ;

        while (polozkyRs.next()) {
            Polozka polozka = new Polozka().setId(polozkyRs.getInt("ID")).setNazev(polozkyRs.getString("nazev"))
                .setCena(polozkyRs.getDouble("cena")).setDruh(polozkyRs.getString("druh"))
                .setActive(polozkyRs.getBoolean("isActive"));
            polozky.add(polozka);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return polozky;
}
```


## Administrace

Administraci provádí klient Admin. Při spuštění se otevře okno s rozcestím, kde jsou dva JButtony. Button s nápisem Nabízené produkty zobrazí okno s vypsanými položkami a přídavkami z databáze. Druhy s nápisem Administrace otevře okno AdministraceRozcesti, kde jsou Buttony s úpravou položky nebo přídavku a vložení položky nebo přídavku .



### Vložení přídavku či položky

Při kliknutí na button vložení se otevře Frame ve kterém jsou pole pro zapsání názvu a ceny a u položky ještě je JComboBox s druhem položky. A Button s potvrzením vložení. Při zmáčnutí buttonu potvrzení se pomocí Naming.lookup odkaže na Třidu na serveru vytvoří se objekt a nastaví se mu hodnoty, které se vezmou z daných polí ve Framu. Při úspěšném zapsání se objeví JOptionPane s oznámením že daná položka nebo přídavek se zapsal.



—

□

×

lamberger

▼

Vložit

### Úprava položky či přídavku

Při kliknutí na úpravu se otevře okno se všemi položkami nebo přídávky. Při kliknutí na položku nebo přídavek se otevře okno, kterému v konstruktoru předávám tu danou položku na kterou jsem klikl, s formulářem, kde jsou hodnoty té dané položky/ přídavku, který chci upravovat. Při potvrzení úpravy se vytvoří ten daný objekt, nastaví se mu hodnoty a předá se metodě, která má za úkol upravit data v databázi. Pokud vše proběhlo v pořádku, zobrazí se hláška, že položka/ přídavek byla upravena.



## Uložiště

### Shared.Uloziste

Dědí od třídy Remote

Rozhraní obsahuje metody:

```
1. public boolean zapisDoUloziste(Objednavka objednavka)
2. public List<Objednavka> vratObjednavky()
```

### Třída Shared.Uloziste

Zde v konstruktoru této třídy vytvářím timer, který bude každou hodinu volat metodu Automaticke mazání, která projde zdejší list objednávek a odstraní objednávky starší než 10 minut.



```

src > shared > Uloziste.java > ...
10 import java.util.TimerTask;
11
12 You, a day ago | 1 author (You)
13 public class Uloziste extends UnicastRemoteObject implements IUloziste {
14     /**
15      *
16      */
17     private static final long serialVersionUID = -5463203417346866089L;
18     private LinkedList<Objednavka> listObjednavek = new LinkedList<>();
19
20     public Uloziste() throws RemoteException {
21
22         Timer timer = new Timer();
23         You, a day ago | 1 author (You)
24         TimerTask hodina = new TimerTask() {
25             @Override
26             public void run() {
27                 AutomatickeMazani();
28             }
29         };
30         timer.schedule(hodina, 01, 1000 * 60 * 60);
31     }
32
33     @Override
34     public List<Objednavka> vratObjednavky() throws RemoteException {
35         return this.listObjednavek;
36     }
37
38     @Override
39     public boolean zapisDoUloziste(Objednavka objednavka) {
40         if (this.listObjednavek.add(objednavka))
41             return true;
42         else
43             return false;
44     }
45
46     private void AutomatickeMazani() {
47         if (!this.listObjednavek.isEmpty()) {
48             for (Objednavka objednavka : listObjednavek) {
49                 Instant casObj = Instant.parse(objednavka.getCasObjednavky());
50                 Instant now = Instant.now();
51
52                 if (Duration.between(casObj, now).compareTo(Duration.ofMinutes(10)) == 1) {
53                     this.listObjednavek.remove(objednavka);
54                 }
55             }
56         }
57     }
58 }

```

## E-R diagram

