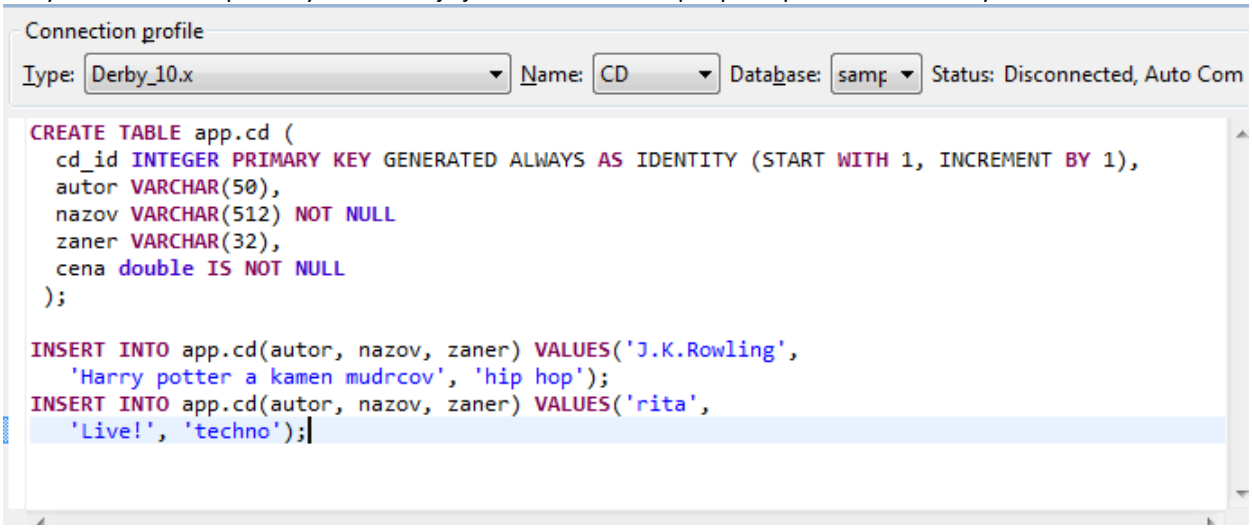
Vytvorila som si project, pomocou návrhového vzoru Visitor, kde som si definovala rozhranie VisitorCd, ktore mi umožňuje pristupovať pomocou metody visit k Akciovym CD a CD, ktore su novinky. Umožňuje nám zistiť maximálnu cenu a cenu týchto cdčiek. Ked si vyžiada cenu CD, ktore je v akcii zisti ju a urci podla nej, či je CD v akcii a ak je táto cena vyššia ako maximálna cena zase ju vyžiada a na základe prijatej metody určí že ide o CD, ktore su novinky, lebo su drahsie. Zistuje udaje z databazy. Dalej obsahuje triedu ZanreVisitor, ktore umožnuje jednotlivým volam specificku metodu, ktora zistiju zanre a na základe nich pridava hodnoty. Obsahuje dve metoda jedna je volana pre akcioveCd a jedna pre NovinkyCD. Potom som si vytvorila Zoznam mojichCD a tabulku DAO CD v ktorej su ulozene vsetky SQL dotazy CD, z ktorych budem hodnoty pridavat, vyberat a aj vymazat. Pristupujem k nim cez DAOCD co predstavuje rozhranie a pristup k jdbc relacnym databazovym objektom. Triedy Servlet je súcast MVC modelu, v ktorom sa najprv prijme požiadavky na html stranku, kde kontajner obdrzi spravu posle ju web brovseri, kontajner najde spravny servlet podla url, ktory je v nom definovany, a vola triedu ZanerVisitor, ktora odosle odpoved na stranku jsp.

Vytvorenie návrhového vzoru javy CDDAO ako slúžiaci na separáciu objektu a prístupu k jednotlivým datam, ktoré sú vytvorené JDBC sql dotazy. Predstavuje jednotné rozhranie pre prístup k dátam z relačných databáz.




```
package DAO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Logger;

import databaza.DBManager;
import databaza.DabazaCD;
import databaza.DatabazaVynimka;
/**
 * Vytvorenie DAO triedy, ktorá predstavuje vytvorenie tabuľky CD,
 * z ktorej vyberám hodnoty pomocou príkazu Select,
 * vymažem pomocou Delete,
 * a vložím pomocou príkazu Insert into
 * Tabuľka má hodnoty, ktoré sa priradia jednotlivým CD-čkam,
 * ktoré sa budú vkladať, ID CD ako primarny kluc,
 * nazov cd, meno autora CD, a zaner
 * JAVA CDDAO(Data Access Object) trieda mi umožňuje prístu k dátam z
 * relačných databáz, umožňuje načítač výsledky JDBC,
 * služi na separaciu objektov čím sa potom uľahčuje prístup k nim ako k celku a
taktiez aj k
 * jednotlivým riadkom SQL dotazov
```

```java
 * @author Sveta
 *
 */
public class CDDAO
{
		private final Logger logger =
Logger.getLogger(this.getClass().getName());
		private final static String SQL_ALL_VALUES = "SELECT * FROM cd";
		private final static String SQL_DELETE_VALUE = "DELETE FROM cd WHERE
id=?";
		private final static String SQL_INSERT_VALUE = "INSERT INTO cd (id,
nazov, autor, zaner, cena) values (default, ?, ?, ?, ?, ?)";
		DabazaCD database;
/**
 * @this.database spojenie s relačnou databazou, vytvori inštanciu CD
 */
public CDDAO() {
	this.database = DBManager.getDatabase();
}
/**
 * Connection - vytvorenie pristupu k databaze
 * Statement - vyhlasenie, že čaká
 * ResultSet - vykovanie prikazov
 * executeQuery - metoda, ktorá slúži k posielaniu dotazov do databazy SQL,
 * výsledkom sú objekty typu ResultSet ako zoznam Stringov primitívnych dátových
typov,
 * pre každý stlpec
 * @return stringList
 */
public List<String> selectAllValues() {
	Connection connection = this.database.getConnection();
	Statement stmt = null;
	ResultSet rs = null;
	try {
		stmt = connection.createStatement();
		rs = stmt.executeQuery(SQL_ALL_VALUES);

		List<String> stringList = new ArrayList<String>();
		while (rs.next()) {

			stringList.add(toString(rs));
		}
		return stringList;

	} catch (SQLException ex) {
		this.logger.severe("Nastala chyba: " + ex.getMessage());
		throw new DatabazaVynimka("Nastala chyba.", ex);
	} finally {
		this.database.closeAll(rs, stmt, connection);
	}
}
/**
 * Vlozenie atributov cd sql dotazov pre entitu CD
 * @param nazov výsledkom je String nazov CD
 * @param autor výsledkom je String meno autora
 * @param zaner výsledkom je String zaner
 * @param cena výsledkom je cena CD, uchovavajuca hodnoty typu double
 * @return
 */
```

```java
public boolean insertValue(String nazov, String autor, String zaner,
		double cena) {
	Connection connection = this.database.getConnection();
	PreparedStatement stmt = null;
	try {
		stmt = connection.prepareStatement(SQL_INSERT_VALUE);
		stmt.setString(1, nazov);
		stmt.setString(2, autor);
		stmt.setString(3, zaner);
		stmt.setDouble(4, cena);

		int addedRows = stmt.executeUpdate();

		return addedRows > 0;

	} catch (SQLException ex) {
		this.logger.severe("Nastala chyba: " + ex.getMessage());
		throw new DatabazaVynimka("Nastala chyba.", ex);
	} finally {
		this.database.closeAll(null, stmt, connection);
	}

}
/**
 * DeleteValue - príkaz, ktorý vymaže danú hodnotu z tabuľky
 * @param id
 * @return vymaze riadky z SQL dotazu
 * closeAll - uzavrie spojenie s databazou
 */
public boolean deleteValue(String id) {
	Connection connection = this.database.getConnection();
	PreparedStatement stmt = null;
	try {
		stmt = connection.prepareStatement(SQL_DELETE_VALUE);
		stmt.setString(1, id);
		int deletedRows = stmt.executeUpdate();

		return deletedRows > 0;

	} catch (SQLException ex) {
		this.logger.severe("Nastala chyba: " + ex.getMessage());
		throw new DatabazaVynimka("Nastala chyba.", ex);
	} finally {
		this.database.closeAll(null, stmt, connection);
	}
}
/**
 * String toString vracia výsledky vykonaných Sql dotazov
 * @param rs
 * @throws SQLException
 * @return výsledkom je tabulka obsahujuca stlpce ID, nazov, Autor, Cena
 */
public String toString(ResultSet rs) throws SQLException {
	String retId = rs.getString(1);
	String retNazov = rs.getString(2);
	String retAutor = rs.getString(3);
	String retZaner = rs.getString(4);
	double retCena = rs.getDouble(5);
```

```
        return "<tr><td>" + retId + "</td><td>" + retNazov + "</td><td>"
                        + retAutor + "</td><td>" + retZaner + "</td><td>" + retCena
                        + "</td><td><td><a href='/Projekt/Vypisat?idVymaz=" + retId
                        + "'>vymaž</a></td></td>";
}
}
```

```
package databaza;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
public interface DabazaCD
{
        /**
         * @autor Sveta
         * Vytvorenie všeobecného rozhrania pre databazu CD, umožňuje prístup k ľubovoľnej databáze,
         *
         */
                        /**
                         * Vytvorenie konektora pre pripojenie k JdbC
                         * @return
                         */
                        public Connection getConnection();

                        /**Zatvori pustupne vsetky objekty, ktore pracovali s Databazou,
                         * @param rs {@link ResultSet}
                         * @param stmt {@link Statement}
                         * @param con {@link Connection}
                         */
                        public void closeAll(ResultSet rs, Statement stmt, Connection con);



        }
```

```
package databaza;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Logger;
public class DatabazaDerby implements DabazaCD
{
/**
 * Vytvori uz konktretnu implementaciu rozhrania DatabazaCD pre DatabazuDerby,
```

```
* nadvezuje spojenie s databazou, ,ktoré je identifikované pomocou
* @url "jdbc:derby://localhost:1527",
* potom identifikuje nazov ovladaca Derby a snim spojene hodnoty,
* pri vytvarani DB ako nazov DB, pouzivatel a heslo
* @autor Sveta
*/
```

```
private static final String ULR = "jdbc:derby://localhost:1527";
private static final String DRIVER_NAME = "org.apache.derby.jdbc.ClientDriver";
private final Logger logger = Logger.getLogger(this.getClass().getName());
private String databaseName;
private String user;
private String password;

/**
 * Vytvori kopiu DatabazyDerby, ktoru sme vytvorili a jej meno je sample,
 * pouzivatel user, a password ako heslo
 */
public DatabazaDerby(String user, String password) {
	this(user, password, "sample");
}

/**
 * Vytvori instaciu databazy s  menom  (databaseName).
 * pomocou Class.forName nahrame a registruje ovladac(driver) databazy, ak sa nenašiel
 * vyhodní chybu (Driver sa nenašiel) ešte pred tym ako
 * budu spravene prve operacie s datami
 */
public DatabazaDerby(String user, String password, String databaseName) {
	this.databaseName = databaseName;
	this.user = user;
	this.password = password;
	this.logger.info("Vytvorena databaza: " + databaseName);
	try {
		Class.forName(DatabazaDerby.DRIVER_NAME);
	} catch (ClassNotFoundException e) {
		this.logger.severe("Driver sa nenasiel");
		throw new DatabazaVynimka("Driver pre DB Derby sa nenasiel");
	}
}

/**
 * Zatvorenie spojenia so vsetkymi objektami, ak sa nepodari vyhlasi chybu
 * (neporadilo sa zavriet )
 */
@Override
public void closeAll(ResultSet rs, Statement stmt, Connection con) {
	try {
		if (rs != null) {
			rs.close();
		}
		if (stmt != null) {
			stmt.close();
		}
		if (con != null) {
```

```
                                        con.close();
                                }
                        } catch (SQLException ex) {
                                this.logger.severe("Nepodarilo sa zavriet.");
                        }
                }

                /**Snazi sa vytvorit spojenie s databazou, konekciu
                 * ak sme autorizovani k ziskaniu pripojenia metoda getConnection
                 * vrati pozadovanie spojenie a objekt ak nie tak
                 * @throws DatabazaVynimka - v pripade, ze nieco pri vytvarani konekcie sa pokazilo
                 */
                @Override
                public Connection getConnection() {
                        try {
                                String url = getDatabaseUrl();
                                Connection connection = DriverManager.getConnection(url, this.user,
                                                this.password);
                                return connection;
                        } catch (SQLException ex) {
                                this.logger.severe("Nepodarilo sa pripojit.");
                                throw new DatabazaVynimka("Nepodarilo sa pripojit.", ex);
                        }
                }

                private String getDatabaseUrl() {
                        return DatabazaDerby.ULR + "/" + this.databaseName;
                }

        }
```

```
package databaza;



        /**
         *Vytvorenie triedy DatabazaVynimka, ktoré dedí od svojej nadtriedy
         *RuntimeException ako nekontrolovana výnimka v príprade vzniku chyby v
databaze
         *@autor Sveta
         */
        public class DatabazaVynimka extends RuntimeException {
                /**
                 * Vytvori kopiu triedy DatabazaVynimka vracajuca hodnoty message a
source,
                 * ktore su zdenene od nadtriedy Throwable
                 */
                public DatabazaVynimka(String message, Throwable source) {
                        super(message, source);
                }

                /**
                 * Vytvori instanciu chyby s nazvom message
                 * @param message - vrati jeden alebo viac SQL errorov, message popisuje
súčasnú výnimku
                 *
                 */
                public DatabazaVynimka(String message) {
                        super(message);
```

```
		}
	}
```

```
package databaza;

public class DBManager
{

	/**
	 * Vytvorenie Triedy, ktora sluzi ako manazer pre vytvorenu Databazu Cd, ktora ju môze riadit
	 * je vseobecna trieda pre vsetky databazy ale sucasne k nej moze pristupovat iba prave
	 * moja vytvorena a pouzivana databa CD
	 * @autor Sveta
	 *
	 */
		private static DabazaCD database;

		/**
		 * Inicializuje databazu, ktoru vytvara Databaza CD
		 */
		public static void registerDatabase(DabazaCD database) {
			DBManager.database = database;
		}

		/**
		 *DatabazaCD vrati kopiu vytvorenej Databazy ak je splnená podmienka if
		 *inak vyhodi nekontrolovanu vynimku
		 */
		public static DabazaCD getDatabase() {
			if(database == null){
				throw new IllegalStateException("DB manager neni vytvoreni");
			}
			return database;
		}

	}
```

Vytvorenie návrhového vzoru Visitor, ktorý umožňuje rozširovať možnosti objektu bez potreby modifikácie jeho triedy. Umožňuje pre danú skupinu tried AkciaCD a NovinkyCD nove operácie. Vytvara 2 typy tried – triedy, ktoré sú navštevované a triedy, ktoré navštevujú.

```
package MojeTriedy;
/**
 * Vytvorenie interface CDVisitor
 * Toto rozhranie implementuje vsetky navstevované triedy
 * v tomto príprade je to Trieda NovinkyCD a AkciaCD
 * deklaruje operáciu Visit pre každú triedu CD v štruktúre objektov.
 * Meno operácie a jej argumenty určujú triedu,
 * ktorá posiela požiadavku Visit triede Visitor.
 * @author Sveta
 *
 */
public interface CDVisitor
{

		public abstract void visit(NovinkyCD n);
```

```
                    public abstract void visit(AkciaCD a);

                    }
```

```
package MojeTriedy;
/**
 * Vytvorenie abstraknej Triedy CD, ktorej zoznam atributov je pomocou pristupového
 * modifikatoru protected to znamena, ze nemozme vytvorit ich instanciu, su urcene na
dedenie
 * a metody su public
 * vymenuva svoje atributy,
 *
 * ObjektCD -CD je navstevovana trieda, obsahuje metodu accept, ktora navstevnika
prijme
 * a da mu data pre pracu
 * @author Sveta
 *
 */
public abstract class CD
{
           protected int ID;
           /**
            * Vytvori instaciu pre Triedu CD, ktorej priradi hodnotu ID
            * @param ID, ukazovatel na sucasnu instanciu pomocou this k datovej
premennej,
            * nema k nim pristup lebo su typu protected
            */
           public CD(int ID)
            {this.ID = ID;}
           public int getID()
              { return ID; }
           public void setID(int ID) {this.ID = ID; }
             protected String nazov;
             public CD(String nazov)
             {this.nazov = nazov;}
             public String getNazov()
             { return nazov; }
             public void setNazov(String nazov) { this.nazov = nazov; }
             public  String autor;
             public void setAutor(String autor) {this.autor = autor;}
             public String getAutor() {return autor;}
             protected String zaner;
             public void setZaner(String zaner) { this.zaner = zaner;}
             public String getZaner()
             {return zaner;}
             protected double cena;
             public abstract double getCena();
             public void setCena(double cena) {this.cena = cena;}
             public String toString()
             {return
"<tr><td>"+ID+"</td><td>"+nazov+"</td><td>"+autor+"</td><td>"+zaner+"</td>"+cena+"</t
r>";}
              /**
               * Keď  akceptuje daný objekt Visitor (t.j. v metóde Accept), pošle
požiadavku objektu VisitorCD,
               * v ktorej sa nachadza
               * @param c
               */
             public abstract void accept(CDVisitor c);
```

```
            }
```

```
package MojeTriedy;
/**
 * Vytvorenie triedy AkciaCD, ktora dedi od nadtriedy CD
 * @author Sveta
 *Odvodena trieda, ktora v metode accept vola naspet metodu objektu
 *CDVisitor
 *Predstavuje navstevovanu Triedu
 */
public  class AkciaCD extends CD
{
            protected double cena;
            /**
             * Vytvorenie instancie obsahujucej zoznam atributov zdenenych z
nadtriedy
             * @param string
             * @param nazov je zdeneny od natriedy pomocou metody super a vracia
jeho hodnotu
             * @param zaner
             * @param cena
             */
            public AkciaCD(String string, String nazov,String zaner, double cena) {
            super(nazov);
            this.cena = cena;
            }
      /**
       * Urobi operaciu setCena pomocou this
       */
            public void setCena(double cena) { this.cena = cena; }
            public double getCena() { return cena; }
            /**
             *  Prijmi daneho visitora (metoda s premennym poctom parametrov, do
metody
             *  @param cd visitor
             *  @param params parametry volaneho visitora
             */
            public void accept (CDVisitor c) { c.visit(this); }
            }
```

```
package MojeTriedy;


import java.util.Enumeration;
import java.util.Vector;
/**
 *  Vytvorenie triedy NovinkyCD, ktora dedi od nadtriedy CD
 * @author Sveta
 *Odvodena trieda, ktora v metode accept vola naspet metodu objektu
 *CDVisitor
 *Predstavuje navstevovanu Triedu, ktora prijme accept metodu
 * @author Sveta
 *
 */
public class NovinkyCD extends CD
{
      /**
```

```
        * Definovanie kontajnera primitivnych datovych typov obsahujuci genericky
datovy typ CD
        * aby sme vedeli aku hodnotu tam budeme vkladat a vytvori premennu cdcka ako
pole Stringu
        */
       protected Vector<CD> cdcka;
       /**
        * Vytvorenie aktualnej instancie triedy NovinkyCD vracujucej hodnotu nazov od
        * jej nadtriedy CD, ktora je implementovana pomocou tejto metody
        * @param nazov
        */
       public NovinkyCD (String nazov) {
       super(nazov);
       cdcka = new Vector<CD>();
       }
       /**
        * vlozenie parametra c do kontajnera Vektor obsahujuci zoznam cdciek
        * @param c
        */
       public void addCD (CD c) {
       cdcka.addElement(c);
       }
/**
 * vymazanie parametra c predstavujuceho cdcko
 * @param c
 */
public void removeCD (CD c) {
cdcka.removeElement(c);
}

public double getCena() {
double totalCena = 0.0;
Enumeration<CD> e = cdcka.elements();
while (e.hasMoreElements()) {
totalCena += ((CD) e.nextElement()).getCena();
}
return totalCena;
}
/**
 * implementacia metody accept c
 */
public void accept (CDVisitor c) { c.visit(this); }
}
```

```
package MojeTriedy;

import java.util.ArrayList;
import java.util.List;

public class ZanerVisitor implements CDVisitor{
        public ZanerVisitor() { }
        /**
         * Metoda predstavujuca konkretnu akciu visit, ktoru dany navstevnik poskytuje z instacie
         * CDVisitor
         * parameter a, na ktorom sa prevedia konkretna akcia
         */
        public void visit (AkciaCD a) {
                /**
                 * Vytvorenie noveho zoznamu zanrov obsahujucich genericke typy
                 * String vracia hodnoty ktore su volane z nadtriedy
```

```
                    * java.lang.Object equals
                    * prida hip-hop a zanre medze ne
                    */
                    List <String> zaner  = new ArrayList<String>();

                if (zaner.equals("hip-hop")) {
                  zaner.add("slovensky");
                  zaner.add("americky");
                }
                else if (zaner.equals("techno")) {
                  zaner.add("Dj");
                  zaner.add("mix");
                }          }
            /**
             * Metoda predstavujuca konkretnu akciu visit, ktoru dany navstevnik poskytuje z instacie
             * CDVisitor
             * Vytvorenie noveho zoznamu zanrov obsahujucich genericke typy
             * String vracia hodnoty ktore su volane z nadtriedy
             * java.lang.Object equals**/
public void visit (NovinkyCD n) {
            List<String> zaner = new ArrayList<String>();

    if (zaner.equals("hip-hop")) {
      zaner.add("slovensky");
      zaner.add("americky");
    }
    else if (zaner.equals("techno")) {
      zaner.add("EN");
      zaner.add("br");
}
}
/**
 * Vracia zoznam zanrov ako pole Stringov, vracia prazdny zoznam
 * @param c
 * @return
 */
public List<String> getZaner(String c)
{
          return null;
}

}
```

```
package MojeTriedy;
/**
 * Vytvorenie triedy, impelementujucej akcie z rozhrania CDVisitor,
 * aby boli vytvorene triedy zobrazne vo vystupe, umoznuje vypis dat
 * Vola operaciu visitor, ktora prislucha tej ktorej triede (akcia a) alebo (novinky
cd)
 * Je aj argumentom sam sebe.**/
public  class ZoznamVisitor implements CDVisitor
{
              public ZoznamVisitor() {}
              public void visit (AkciaCD a) {
              System.out.println("Zoznam CD v akcii");
              }
              public void visit (NovinkyCD n) {
              System.out.println("Zoznam CD novinike");
              }
```

```
		}
```

```
package MojeTriedy;
/**
 * Deklaracia hlavnej triedy vyis, ktora testuje maximalnu cenu cd v akcii a noviniek
CD
 * @author Sveta
 *
 */
public class Vypis
{
		public static void main (String[] args) {
			/**
			 * Deklaracia cd V akcii, vytvorenie instancie k AkciaCD a naplni
parametre svojimi
			 * novymi atributmi.
			 */
		AkciaCD a1 = new AkciaCD("hityLeto","kazik", "rap", 10.00);
		AkciaCD a2 = new AkciaCD("hity zima","ja", "ludovka", 20.00);
		/**
		 * Vytvorenie instancie NovinkyCD a postupne priradovanie hodnot
		 *
		 */
		NovinkyCD n = new NovinkyCD("Char");
		n.addCD(a1);
		n.addCD(a2);
		/**
		 * Volanie metody visit jej akceptovanie z triedy ZoznamVisitor
		 */
		ZoznamVisitor zv = new ZoznamVisitor();
		a1.accept(zv);
		a2.accept(zv);
		n.accept(zv);
		/**
		 * Vytvorenie instancie, ktora robi konrektnu akciu, akceptovanie metody
		 * z triedy CenaVisitor
		 */
		CenaVisitor cv = new CenaVisitor(25.00);
		a1.accept(cv);
		a2.accept(cv);
		n.accept(cv);
}
}
```

```
package Servlet;
import java.io.IOException;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import DAO.CDDAO;
import databaza.DBManager;
import databaza.DatabazaDerby;
public class CD_v extends HttpServlet
{

/**
 * Implementacia Servletu
 */
```

```
		private static final long serialVersionUID = 1L;
		private DatabazaDerby db;

		public CD_v() {
			db = new DatabazaDerby("user", "user");
			DBManager.registerDatabase(db);
		}

		/**
		 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
		 *     response)
		 *     Vytvorenie instancie CDDAO, ktora je volana z CDDAO
		 */
		protected void doGet(HttpServletRequest request,
				HttpServletResponse response) throws ServletException, IOException {
			CDDAO CdDao = new CDDAO();

			String idVymaz = request.getParameter("idVymaz");

			Integer id = 0;

			if (idVymaz != null)
				id = Integer.parseInt(idVymaz);

			if (id > 0) {
				CdDao.deleteValue(idVymaz);
			}

			List<String> values = CdDao.selectAllValues();

			request.setAttribute("list", values);
			request.setAttribute("id", idVymaz);
			request.getRequestDispatcher("vypis.jsp").forward(request, response);
		}
	}
```

```
package Servlet;
/**
 * @autor Sveta
 * Deklaracia abstraktnej triedy MyCD, ktora dedi od svojej nadtriedy
 * CD
 */
import MojeTriedy.CD;

public abstract class MyCD extends CD
{
		public MyCD(int ID)
	{
		super(ID);
	}

		private String nazov = "";

		public void setNazov(String nazov) {
			this.nazov = nazov;
		}

		public String getNazov() {
			return nazov;
```

```
	}

	private String autor = "";

	public void setAutor(String autor) {
		this.autor = autor;
	}

	public String getAutor() {
		return autor;
	}

	private String zaner= "";

	public void setZaner(String zaner) {
		this.zaner = zaner;
	}

	public String getZaner() {
		return zaner;
	}

	private double cena = "";

	public void setCena(double cena) {
		this.cena = cena;
	}

	public double getCena() {
		return cena;
	}


}
```

```
package Servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.derby.database.Database;

import MojeTriedy.CD;
import DAO.CDDAO;
import MojeTriedy.CDVisitor;

import databaza.DBManager;
import databaza.DatabazaDerby;
public class MyFormServlet extends HttpServlet
{

	/**
	 * Servlet, ktory implementuje MyFormSerlvet
	 */
		private static final long serialVersionUID = 1L;
		private DatabazaDerby db;
```

```
public MyFormServlet() {
		db = new DatabazaDerby("user", "user");
		DBManager.registerDatabase(db);
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
 *      response)
 */
protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
		doPost(request, response);
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 *      response)
 */
protected void doPost(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
		request.setCharacterEncoding("utf-8");
		MyCD cd= new MyCD ();
		String nazov = request.getParameter("nazov");
		String autor = request.getParameter("autor");
		String zaner = request.getParameter("zaner");
		Double cena = request.getParameter("cena");
		String odoslane = request.getParameter("odoslane");
		String uNazov = "";
		String uAutor = "";
		String uZaner= "";
		String uCena = "";
		boolean jeOdoslane = false;
		if (odoslane != null) {
			jeOdoslane = true;
			if ((vyplnene(nazov)) && (ibaPismena(nazov)))
				uNazov = "";
			else if (!vyplnene(nazov))
				uNazov = "Údaj je povinný";
			else if (!ibaPismena(nazov))
				uNazov = "Nesprávne zadaná hodnota";
			if (ibaPismena(autor))
				uAutor = "";
			else
				uAutor = "Nesprávne zadaná hodnota";
			try {
				if (!ibaCisla(zaner))
					throw new ZadaniePismenaException();
				uZaner = "";
			} catch (ZadaniePismenaException ex) {
				uZaner = ex.getMessage();
			}
			try {
				if (!ibaCisla(cena))
					throw new ZadaniePismenaException();
				uCena = "";
			} catch (ZadaniePismenaException ex) {
				uCena = ex.getMessage();
```

```
                }

            }
            cd.setNazov(nazov);
            cd.setAutor(autor);
            cd.setZaner(zaner);
            cd.setCena(cena);
            request.setAttribute("cd", cd);
            request.setAttribute("uNazov", uNazov);
            request.setAttribute("uAutor", uAutor);
            request.setAttribute("uZaner", uZaner);
            request.setAttribute("uCena", uCena);
            if ((jeOdoslane) && (vyplnene(cd.getNazov()))
                        && (ibaPismena(cd.getNazov())) && ibaPismena(cd.getAutor())
                        && (ibaCisla(cd.getZaner()))  )
                         {
                CDDAO cdDao = new CDDAO();
                cdDao.insertValue(nazov, autor, zaner, cena);
                request.getRequestDispatcher("vloz.jsp").forward(request, response);
            } else
                request.getRequestDispatcher("form.jsp").forward(request,
                                response);
        }



        private boolean ibaCisla(String zaner)
        {
            return false;
        }

        boolean ibaPismena(String s) {
            boolean n = true;
            for (int i = 0; i < s.length(); i++)
                if (Character.isDigit(s.charAt(i))) {
                    n = false;
                    break;
                } else
                    n = true;
            return n;
        }

        boolean vyplnene(String s) {
            if (s.equals(""))
                return false;
            else
                return true;
        }
    }

    class ZadaniePismenaException extends Exception {
        public ZadaniePismenaException() {
            super("Nesprávne zadaná hodnota");
        }
    }
```

```
package Servlet;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;



import javax.servlet.*;
import javax.servlet.http.*;

import MojeTriedy.ZanerVisitor;

import java.io.*;
import java.util.*;
public class VyberServlet extends HttpServlet
{
          /**
           * Metoda doPost
           * Servlet implementujuci VyberServlet obsahuje metodu doPost
           * @param request obsahuje vsetky informacie od triedy ZanerVisitor v podobe Listu Stringov
           * @param response obsahuje už odpoved, ktoru mi poskytne jsp stranka - vyberZaner.jsp
           * @throws IOException
           * @throws ServletException
           */
 public void doPost( HttpServletRequest request,
                          HttpServletResponse response)
                          throws IOException, ServletException {

             String c = request.getParameter("zaner");
/**
 * Vytvorenie instancie ZanerVisitor
 */
             ZanerVisitor vs = new ZanerVisitor();

             List<String> result = vs.getZaner(c);


             request.setAttribute("styles", result);
             RequestDispatcher view = request.getRequestDispatcher("VyberZaner.jsp");
             view.forward(request, response);
           }
          }
```

```
package Servlet;
/**
 * Implementacia servletu CenaServlet
 * je zalozeny na principe model-view-controller, model predstavuje trieda
 * Vypis, Controller predstavuje trieda Servlet, view predstavuje jsp - CenaVypis
 */
import javax.servlet.*;
import javax.servlet.http.*;

import MojeTriedy.CenaVisitor;
```

```
import java.io.*;
import java.util.*;
public class CenaServlet extends HttpServlet
{
        /**
         * Implementovanie metody doPost, kde je vstupnym parametrom cena
         * @param request
         * @param response
         * @throws IOException
         * @throws ServletException
         */
         public void doPost( HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException {

/**
 * Vytvorenie instancie Vypis
 */
            Vypis cv = new Vypis(0);

/**
 * Vysledok volania metody doPost je vrateny spet(ako atribut) stranke cenaVypis.jsp
 */

            request.setAttribute("a", result);
            RequestDispatcher view = request.getRequestDispatcher("CenaVypis.jsp");
            view.forward(request, response);
          }
        }
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="java.util.*" %>

<html>
<body>
<h1 align="center">Ceny akcia a novinky CD</h1>
<p>

<%
  List styles = (List) request.getAttribute("a");
  Iterator it = styles.iterator();
  while(it.hasNext()) {
    out.print("<br>try: " + it.next());
  }
%>

</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
       pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Cdčka</title>
```

```
</head>
<body>
<form action="" method="post">
<table>
	<tr>
		<td>Nazov:</td>
		<td><input type="text" name="nazov" value="${cd.nazov }">*</td>
		<td><span style="color: red;"> ${uNazov }</span></td>
	</tr>
	<tr>
		<td>Autor:</td>
		<td><input type="text" name="autor" value="${cd.autor }"></td>
		<td><span style="color: red;"> ${uAutor}</span></td>
	</tr>
	<tr>
		<td>Zaner:</td>
		<td><input type="text" name="zaner" value="${cd.zaner }"></td>
		<td><span style="color: red;"> ${uZaner }</span></td>
	</tr>
	<tr>
		<td>Cena:</td>
		<td><input type="text" name="cena" value="${cd.cena }"></td>
		<td><span style="color: red;"> ${uCena }</span></td>
	</tr>


	<tr>
		<td><input type="hidden" name="odoslane" value="1" /><input
			type="submit" value="Po†’" /></td>
		<td></td>
		<td></td>
	</tr>
</table>
<br>
Udaje oznacene ao povinne©</form>
<br>
<a href="/Sveta_projekt1/Servlet.CD_v">Vypis vsetkych zaznamov</a>
<br>
<br>

</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
	pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>CD - vloz udaje</title>
</head>
<body>
<span style="color: green;"> Udaje boli vlozene</span>
<table>
	<tr>
		<td width=100>Nazov</td>
		<td>${cd.nazov }</td>
	</tr>
	<tr>
```

```
		<td width=100>Autor</td>
		<td>${cd.autor }</td>
	</tr>
	<tr>
		<td width=100>Zaner</td>
		<td>${cd.zaner }</td>
	</tr>
	<tr>
		<td width=100>Cena</td>
		<td>${cd.cena }</td>
	</tr>

	<tr>
		<td><a href="/Sveta_Projekt1/Servlet.CD_v">CD_v</a></td>
		<td><a href="/Sveta_Projekt1/Servlet.MyFormServlet">form</a></td>
	</tr>
</table>
<br>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="java.util.*" %>

<html>
<body>
<h1 align="center">Vyber zaner</h1>
<p>

<%
  List styles = (List) request.getAttribute("styles");
  Iterator it = styles.iterator();
  while(it.hasNext()) {
    out.print("<br>try: " + it.next());
  }
%>

</body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta http-equiv="Cache-control" content="no-cache" />
<meta http-equiv="Pragma" content="no-cache" />
<title>CD - vsetky zaznamy</title>
</head>
<body>
<table>
	<tr>
		<td width=50>ID</td>
		<td width=100>NAZOV</td>
```

```
            <td width=100>AUTOR</td>
            <td width=100>ZANER</td>
            <td width=75>CENA</td>
      </tr>
      <c:if test="${empty list}">
Prazdna tabulka     </c:if>
      <c:forEach items="${list}" var="CD">
            ${mycd}
      </c:forEach>
</table>
<br></br>
<a href="/Sveta_projekt1/Servlet.MyFormServlet">Form</a>
<br></br>
<br></br>
</body>
</html>
```

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

  <servlet>
    <servlet-name>CenaServlet</servlet-name>
    <servlet-class>Servlet.CenaServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>CenaServlet</servlet-name>
    <url-pattern>/CenaServlet.do</url-pattern>
  </servlet-mapping>

</web-app>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>Sveta_Projekt1</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>vypis</display-name>
    <servlet-name>vypis</servlet-name>
    <servlet-class>vypis</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>vypis</servlet-name>
    <url-pattern>/vypis</url-pattern>
  </servlet-mapping>
```

```
  <servlet>
    <description></description>
    <display-name>MyFormServlet</display-name>
    <servlet-name>MyFormServlet</servlet-name>
    <servlet-class>Servlet.MyFormServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyFormServlet</servlet-name>
    <url-pattern>/MyFormServlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <description></description>
    <display-name>CD_v</display-name>
    <servlet-name>CD_v</servlet-name>
    <servlet-class>servlet.CD_v</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CD_v</servlet-name>
    <url-pattern>/CD_v</url-pattern>
  </servlet-mapping>
</web-app>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Zanre Hudby</title>
<body>
  <h1 align="center">Zanre </h1>
  <form method="POST" action="SelectCoffee.do">
    Select zaner
    Type:
    <select name="type" size=1>
      <option value="hip-hop">Hip-hop</option>
      <option value="techno">Techno</option>
    </select>
    <br><br>
    <center>
      <input type="Submit">
    </center>
   </form>
</body>
</html>
```

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

  <servlet>
    <servlet-name>VyberServlet</servlet-name>
    <servlet-class>Servlet.VyberServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>VyberServlet</servlet-name>
    <url-pattern>/VyberServlet.do</url-pattern>
  </servlet-mapping>
```

```
</web-app>
```

```
<Context>
   <Resource name="jdbc/sampleDB" auth="Container"
        type="javax.sql.DataSource"
        username="app" password="app"
        driverClassName="org.apache.derby.jdbc.ClientDriver"
        url="jdbc:derby://localhost:1527/sample"
        maxActive="8"    />
 </Context>
```