|  |
| --- |
| **Pakages in Java**  import com.example.unitconverter.userDao.DAOController;  import javafx.application.Application;  import javafx.fxml.FXMLLoader;  import javafx.scene.Scene;  import javafx.scene.layout.AnchorPane;  import javafx.scene.layout.VBox;  import javafx.stage.Stage;  import java.io.IOException; |

|  |
| --- |
| **Energy code** |

|  |
| --- |
| **BritishThermalUnitConverter.java**  public class BritishThermalUnitConverter extends BaseUnitConverter {  public BritishThermalUnitConverter()  {  this.MULTIPLIER = new BigDecimal(1055.06);  }  @Override  public String getName() {  return "British Thermal Unit";  }  }  **EnergyConverter.java**  public class EnergyConverter extends UnitTypeConverter {  private static BaseUnitConverter jouleConverter = new JouleConverter();  private static BaseUnitConverter kilojouleConverter = new KilojouleConverter();  private static BaseUnitConverter kilocalorieConverter = new KilocalorieConverter();  private static BaseUnitConverter gramCalorieConverter = new GramCalorieConverter();  private static BaseUnitConverter britishThermalUnitConverter = new BritishThermalUnitConverter();  private static BaseUnitConverter wattHourConverter = new WattHourConverter();  public EnergyConverter() {  super("Joule",new HashMap<String,BaseUnitConverter>()  {  {  put(jouleConverter.getName(), jouleConverter);  put(kilojouleConverter.getName(), kilojouleConverter);  put(kilocalorieConverter.getName(), kilocalorieConverter);  put(gramCalorieConverter.getName(), gramCalorieConverter);  put(britishThermalUnitConverter.getName(), britishThermalUnitConverter);  put(wattHourConverter.getName(), wattHourConverter);  }  });  }  @Override  public String getUnitType() {  return "Energy";  }  }  **GramCalorieConverter.java**  public class GramCalorieConverter extends BaseUnitConverter {  public GramCalorieConverter()  {  this.MULTIPLIER = new BigDecimal(4.184);  }  @Override  public String getName() {  return "Gram calorie";  }  }  **JouleConverter.java**  public class JouleConverter extends BaseUnitConverter {  public JouleConverter()  {  this.MULTIPLIER = new BigDecimal(1);  }  @Override  public String getName() {  return "Joule";  }  }  **KilocalorieConverter.java**  public class KilocalorieConverter extends BaseUnitConverter {  public KilocalorieConverter()  {  this.MULTIPLIER = new BigDecimal(4184);  }  @Override  public String getName() {  return "Kilocalorie";  }  }  **KilojouleConverter.java**  public class KilojouleConverter extends BaseUnitConverter {  public KilojouleConverter()  {  this.MULTIPLIER = new BigDecimal(1000);  }  @Override  public String getName() {  return "Kilojoule";  }  }  **WattHourConverter.java**  public class WattHourConverter extends BaseUnitConverter {  public WattHourConverter()  {  this.MULTIPLIER = new BigDecimal(3600);  }  @Override  public String getName() {  return "Watt Hour";  }  }  **CentimeterConverter**  public class CentimeterConverter extends BaseUnitConverter {  public CentimeterConverter()  {  this.MULTIPLIER = new BigDecimal(0.01);  }  @Override  public String getName() {  return "Centimeter";  }  }  **FootConverter.java**  public class FootConverter extends BaseUnitConverter {  public FootConverter()  {  this.MULTIPLIER = new BigDecimal(0.3048);  }  @Override  public String getName() {  return "Foot";  }  }  **InchConverter.java**  public class InchConverter extends BaseUnitConverter {  public InchConverter()  {  this.MULTIPLIER = new BigDecimal(0.0254);  }  @Override  public String getName() {  return "Inch";  }  }  **KilometerConverter.java**  public class KilometerConverter extends BaseUnitConverter {  public KilometerConverter()  {  this.MULTIPLIER = new BigDecimal(1000);  }  @Override  public String getName() {  return "Kilometer";  }  }  **LengthConverter.java**  public class LengthConverter extends UnitTypeConverter {  private static BaseUnitConverter meterConverter = new MeterConverter();  private static BaseUnitConverter footConverter = new FootConverter();  private static BaseUnitConverter centimeterConverter = new CentimeterConverter();  private static BaseUnitConverter kilometerConverter = new KilometerConverter();  private static BaseUnitConverter mileConverter = new MileConverter();  private static BaseUnitConverter millimeterConverter = new MillimeterConverter();  private static BaseUnitConverter inchConverter = new InchConverter();  public LengthConverter() {  super("Meter",new HashMap<String,BaseUnitConverter>()  {  {  put(meterConverter.getName(), meterConverter);  put(footConverter.getName(), footConverter);  put(kilometerConverter.getName(), kilometerConverter);  put(centimeterConverter.getName(), centimeterConverter);  put(mileConverter.getName(), mileConverter);  put(millimeterConverter.getName(), millimeterConverter);  put(inchConverter.getName(), inchConverter);  }  });  }  @Override  public String getUnitType() {  return "Length";  }  }  **MeterConverter.java**  public class MeterConverter extends BaseUnitConverter {  public MeterConverter()  {  this.MULTIPLIER = new BigDecimal(1);  }  @Override  public String getName() {  return "Meter";  }  }  **MileConverter.java**  public class MileConverter extends BaseUnitConverter {  public MileConverter()  {  this.MULTIPLIER = new BigDecimal(1609.344);  }  @Override  public String getName() {  return "Mile";  }  }  **MillimeterConverter.java**  public class MillimeterConverter extends BaseUnitConverter {  public MillimeterConverter()  {  this.MULTIPLIER = new BigDecimal(0.001);  }  @Override  public String getName() {  return "Millimeter";  }  } |

|  |
| --- |
| **Speed**  **KnotConverter.java**  public class KnotConverter extends BaseUnitConverter {  public KnotConverter()  {  this.MULTIPLIER = new BigDecimal(0.514444);  }  @Override  public String getName() {  return "Knot";  }  }  **MeterPerSecondConverter.java**  public class MeterPerSecondConverter extends BaseUnitConverter {  public MeterPerSecondConverter()  {  this.MULTIPLIER = new BigDecimal(1);  }  @Override  public String getName() {  return "Meter / Second";  }  }  **MilesPerHourConverter.java**  public class MilesPerHourConverter extends BaseUnitConverter {  public MilesPerHourConverter()  {  this.MULTIPLIER = new BigDecimal(0.44704);  }  @Override  public String getName() {  return "Miles / Hour";  }  }  **SpeedConverter.java**  public class SpeedConverter extends UnitTypeConverter {  private static BaseUnitConverter meterPerSecondConverter = new MeterPerSecondConverter();  private static BaseUnitConverter milesPerHourConverter = new MilesPerHourConverter();  private static BaseUnitConverter knotConverter = new KnotConverter();  public SpeedConverter() {  super("Meter / Second",new HashMap<String,BaseUnitConverter>()  {  {  put(meterPerSecondConverter.getName(), meterPerSecondConverter);  put(milesPerHourConverter.getName(), milesPerHourConverter);  put(knotConverter.getName(), knotConverter);  }  });  }  @Override  public String getUnitType() {  return "Energy";  }  }  **Temperature:**  **CelsiusConverter.java**  public class CelsiusConverter extends BaseUnitConverter {  public CelsiusConverter()  {  this.MULTIPLIER = new BigDecimal(1);  }  @Override  public String getName() {  return "Celsius";  }  }  **FahrenheitConverter.java**  public class FahrenheitConverter extends BaseUnitConverter {  private static final BigDecimal \_9\_5 = new BigDecimal(1.8);  private static final BigDecimal \_5\_9 = new BigDecimal(0.555556);  private static final BigDecimal \_32 = new BigDecimal(32);  @Override  public BigDecimal convertToBase(BigDecimal value)  {  return value  .subtract(\_32)  .multiply(\_5\_9)  .setScale(4, RoundingMode.HALF\_UP);  }  @Override  public BigDecimal convertFromBase(BigDecimal value)  {  return value  .multiply(\_9\_5)  .add(\_32)  .setScale(4, RoundingMode.HALF\_UP);  }  // public FahrenheitConverter()  // {  // this.MULTIPLIER = new BigDecimal(1);  // }  @Override  public String getName() {  return "Fahrenheit";  }  }  **KelvinConverter.java**  public class KelvinConverter extends BaseUnitConverter {  private final static BigDecimal Kelvin\_constant = new BigDecimal(273.15);  @Override  public BigDecimal convertToBase(BigDecimal value)  {  return value  .subtract(Kelvin\_constant)  .setScale(4, RoundingMode.HALF\_UP);  }  @Override  public BigDecimal convertFromBase(BigDecimal value)  {  return value  .add(Kelvin\_constant)  .setScale(4, RoundingMode.HALF\_UP);  }  @Override  public String getName() {  return "Kelvin";  }  }  **TemperatureConverter.java**  public class TemperatureConverter extends UnitTypeConverter {  private static BaseUnitConverter celsiusConverter = new CelsiusConverter();  private static BaseUnitConverter fahrenheitConverter = new FahrenheitConverter();  private static BaseUnitConverter kelvinConverter = new KelvinConverter();  public TemperatureConverter() {  super("Celsius",new HashMap<String,BaseUnitConverter>()  {  {  put(celsiusConverter.getName(), celsiusConverter);  put(fahrenheitConverter.getName(), fahrenheitConverter);  put(kelvinConverter.getName(), kelvinConverter);  }  });  }  @Override  public String getUnitType() {  return "Temperature";  }  }  **Time**  **DayConverter.java**  public class DayConverter extends BaseUnitConverter {  public DayConverter()  {  this.MULTIPLIER = new BigDecimal(86400);  }  @Override  public String getName() {  return "Day";  }  }  **HourConverter**  public class HourConverter extends BaseUnitConverter {  public HourConverter()  {  this.MULTIPLIER = new BigDecimal(3600);  }  @Override  public String getName() {  return "Hour";  }  }  **MicrosecondConverter.java**  public class MicrosecondConverter extends BaseUnitConverter {  public MicrosecondConverter()  {  this.MULTIPLIER = new BigDecimal(.000001);  }  @Override  public String getName() {  return "Microsecond";  }  }  **MillisecondConverter.java**  public class MillisecondConverter extends BaseUnitConverter {  public MillisecondConverter()  {  this.MULTIPLIER = new BigDecimal(.001);  }  @Override  public String getName() {  return "Millisecond";  }  }  **MinuteConverter**  public class MinuteConverter extends BaseUnitConverter {  public MinuteConverter()  {  this.MULTIPLIER = new BigDecimal(60);  }  @Override  public String getName() {  return "Minute";  }  }  **MonthConverter**  public class MonthConverter extends BaseUnitConverter {  public MonthConverter()  {  this.MULTIPLIER = new BigDecimal(2628000);  }  @Override  public String getName() {  return "Month";  }  }  **NanosecondConverter**  public class NanosecondConverter extends BaseUnitConverter {  public NanosecondConverter()  {  this.MULTIPLIER = new BigDecimal(.000000001);  }  @Override  public String getName() {  return "Nanosecond";  }  }  **SecondConverter**  public class SecondConverter extends BaseUnitConverter {  public SecondConverter()  {  this.MULTIPLIER = new BigDecimal(1);  }  @Override  public String getName() {  return "Second";  }  }  **TimeConverter**  public class TimeConverter extends UnitTypeConverter {  private static BaseUnitConverter minuteConventer = new MinuteConverter();  private static BaseUnitConverter secondConverter = new SecondConverter();  private static BaseUnitConverter hourConverter = new HourConverter();  private static BaseUnitConverter dayConverter = new DayConverter();  private static BaseUnitConverter weekConverter = new WeekConverter();  private static BaseUnitConverter monthConverter = new MonthConverter();  private static BaseUnitConverter yearConverter = new YearConverter();  private static BaseUnitConverter millisecondConverter = new MillisecondConverter();  private static BaseUnitConverter microsecondConverter = new MicrosecondConverter();  private static BaseUnitConverter nanosecondConverter = new NanosecondConverter();  public TimeConverter() {  super("Second",new HashMap<String,BaseUnitConverter>()  {  {  put(secondConverter.getName(), secondConverter);  put(minuteConventer.getName(), minuteConventer);  put(hourConverter.getName(), hourConverter);  put(dayConverter.getName(), dayConverter);  put(weekConverter.getName(), weekConverter);  put(monthConverter.getName(), monthConverter);  put(yearConverter.getName(), yearConverter);  put(millisecondConverter.getName(), millisecondConverter);  put(microsecondConverter.getName(), microsecondConverter);  put(nanosecondConverter.getName(), nanosecondConverter);  }  });  }  @Override  public String getUnitType() {  return "Time";  }  }  **WeekConverter.java**  public class WeekConverter extends BaseUnitConverter {  public WeekConverter()  {  this.MULTIPLIER = new BigDecimal(604800);  }  @Override  public String getName() {  return "Week";  }  }  **YearConverter.java**  public class YearConverter extends BaseUnitConverter {  public YearConverter()  {  this.MULTIPLIER = new BigDecimal(31536000);  }  @Override  public String getName() {  return "Year";  }  } |

|  |
| --- |
| **UserDao**  **DAOController.java**  public class DAOController {  private UserDAO userDAO;  private HelloApplication main;  private Stage primaryStage;  User user;  @FXML  private Label lblPassword;  @FXML  private Label lblUsername;  @FXML  private TextField textPass;  @FXML  private TextField txtUsername;  @FXML  private Label welcomeText;  public String getTxtName() {  return txtUsername.getText();  }  public String getTxtPass() {  return textPass.getText();  }  public void initialize() throws SQLException {  user = new User();  userDAO = new UserDAO();  System.out.println(userDAO);  }  public void setMain(HelloApplication main, Stage primaryStage) {  this.main = main;  this.primaryStage = primaryStage;  }  @FXML  void OnLoginClick() {  String username = getTxtName();  String password = getTxtPass();  boolean result = userDAO.validUser(username, password);  System.out.println("Username: " + username);  System.out.println("Password: " + password);  if (result) {  welcomeText.setText("Validation is successful! Welcome!");  user.setUsername(username);  System.out.println("User: " + user.getUsername());  main.mainWindow();  } else {  welcomeText.setText("User not found!");  }  }  }  **DbUtil.java**  public class DbUtil {  private static Connection connection = null;// 1 connection for the whole app  public static Connection getConnection() {  System.out.println("insoidedbutil");  if (connection != null) {  System.out.println("Your Connection" + connection);  return connection;  }  else {  try {  String driver = "org.postgresql.Driver";  String url = "jdbc:postgresql://localhost:5432/samreen";  String user = "postgres";  String password = "s@me123";  Class.forName(driver);  connection = DriverManager.getConnection(url, user, password);  System.out.println("YOur connection" + connection);  } catch (ClassNotFoundException e) {  e.printStackTrace();  } catch (SQLException e) {  e.printStackTrace();  }  return connection;  }  }  }    **User.java**  public class User {  private int iduser;  private String username;  private String password;  public int getId() {  return iduser;  }  public void setId(int id) {  this.iduser = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public User(int iduser, String username, String password) {  this.iduser = iduser;  this.username = username;  this.password = password;  }  }  **UserDAO.java**  public UserDAO(){  this.connection = DbUtil.getConnection();  System.out.println(connection);  }  public List<User> getAllUsers() throws SQLException {  return null;  }  public void updateUser(User user) {  }  public void deleteUser(int id) {  }  public void addUser(User user) {  try {  PreparedStatement preparedStatement = connection.prepareStatement("insert into users(username,password) values (?, ?, ?, ? )");  // Parameters start with 1  preparedStatement.setString(1, user.getUsername());  preparedStatement.setString(2, user.getPassword());  preparedStatement.executeUpdate();  } catch (SQLException e) {  e.printStackTrace();  }  }  public boolean validUser(String username,String password) {  boolean flag = false;  try {  Statement stmt = connection.createStatement();  System.out.println("connection" + connection);  ResultSet rs = stmt.executeQuery("SELECT \* FROM user WHERE username='" + username+"' and password='"+password+"'");  System.out.println("Your data from Database: " + rs);  if(rs.next()){  flag = true;  }  } catch (SQLException ex) {  ex.printStackTrace();  }    return flag;  }  } |

|  |
| --- |
| **volume** |

|  |
| --- |
| **GallonConverte.java**  public class GallonConverter extends BaseUnitConverter {  public GallonConverter()  {  this.MULTIPLIER = new BigDecimal(3.785);  }  @Override  public String getName() {  return "Gallon";  }  } |