UNIVERSITY OF STIRLING

# CSCU9YW – Assignment Report

Student Number: 250796

## Contents

## 1 Overview

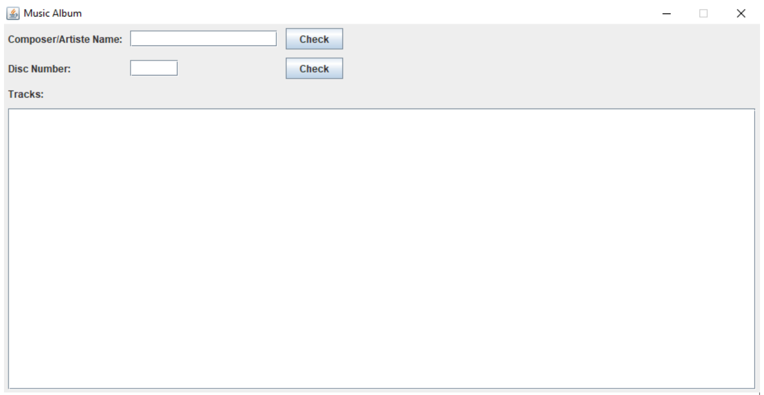
The aim of this project is to use a SOAP web service to search and retrieve music track information from a database.

Using a GUI from the Client class with 2 text fields and buttons of Composer Names and Disc Numbers, the user can enter an integer in the Disc Number field or a string (for the Artist Name) in the Composer Name field.

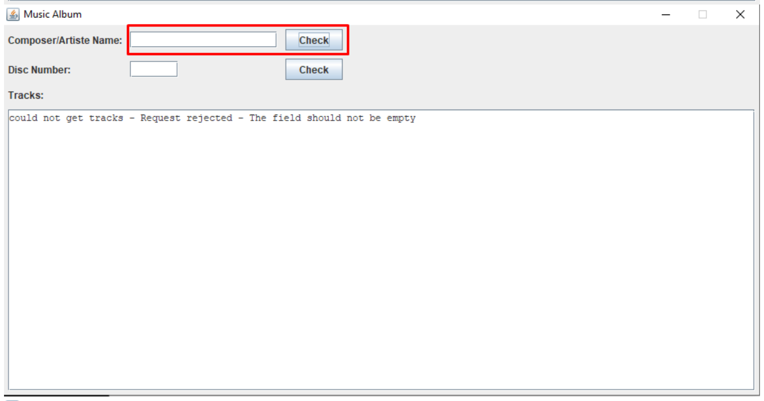
Axis2 is used to convert the WSDL file "Services" to Java (WSDL2JAVA) in order to generate classes that represent client stubs, server skeletons and data types that will help to the writing of the client side of the Web services defined in the WSDL document.

## 2 Solution and Testing

The figure on the right gives an overview of the GUI.

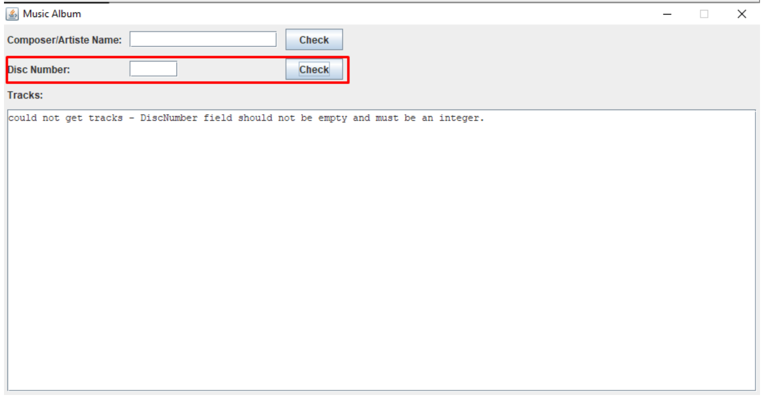


At first, when we check for tracks without input data in the Composer Name field, an error message appears that the field should not be empty.



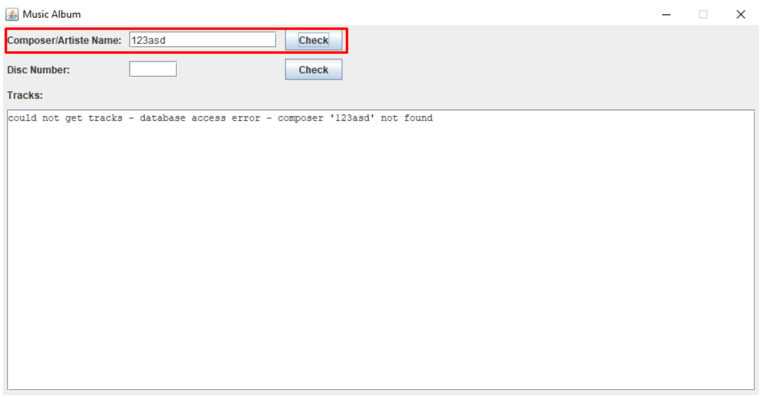
```
could not get tracks - Request rejected - The field should not be empty
```

When we follow the same procedure to the Disc Number (checking for tracks with no input data in the Disc Number field), we get a similar error message. The field should not be empty and should be an integer.



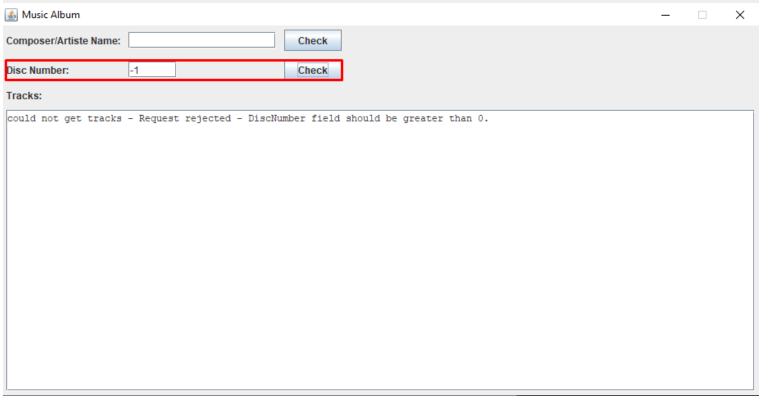
```
could not get tracks - DiscNumber field should not be empty and must be an integer.
```

Then, we tried to check for tracks in the field Composer Name using both number and string input data. As we can see in the figure on the right, it only checks for <u>that name</u> in the database. Therefore, an error message appears that the composer "x" is not found.



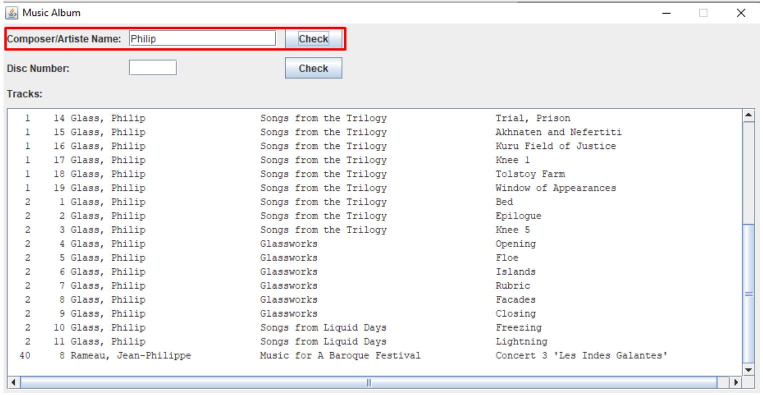
```
could not get tracks - database access error - composer '123asd' not found
```

Since the Disc Number field accepts only integer numbers, we now try to check for tracks with input data of negative integers. In this respect, an error message appears saying that the request is rejected, and the value should be greater than 0. Thus, this field shall be integer and greater that 0 to work.



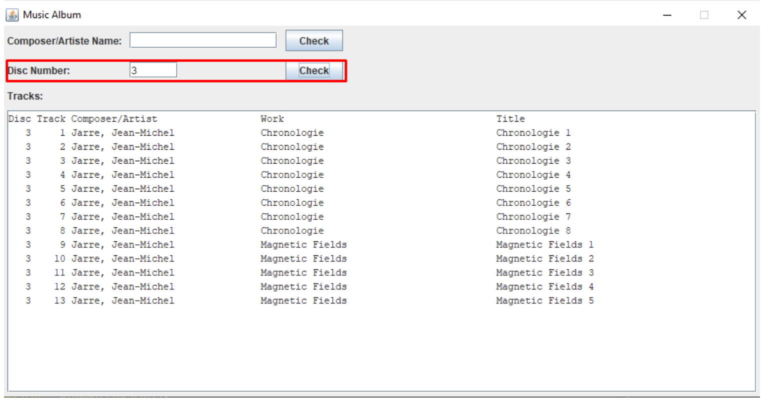
```
could not get tracks - Request rejected - DiscNumber field should be greater than 0.
```

Next, we check for tracks in the Composer Name field with <u>only string input data</u> (e.g. "Philip"), it searches for the name through the database and successfully retrieves all Composer Name Philip data within it. Please note that the service supports partial search, for example, when we search only for “K”, it will return all names that include the letter K.



```
  1    14 Glass, Philip              Songs from the Trilogy          Trial, Prison
  1    15 Glass, Philip              Songs from the Trilogy          Akhnaten and Nefertiti
  1    16 Glass, Philip              Songs from the Trilogy          Kuru Field of Justice
  1    17 Glass, Philip              Songs from the Trilogy          Knee 1
  1    18 Glass, Philip              Songs from the Trilogy          Tolstoy Farm
  1    19 Glass, Philip              Songs from the Trilogy          Window of Appearances
  2     1 Glass, Philip              Songs from the Trilogy          Bed
  2     2 Glass, Philip              Songs from the Trilogy          Epilogue
  2     3 Glass, Philip              Songs from the Trilogy          Knee 5
  2     4 Glass, Philip              Glassworks                      Opening
  2     5 Glass, Philip              Glassworks                      Floe
  2     6 Glass, Philip              Glassworks                      Islands
  2     7 Glass, Philip              Glassworks                      Rubric
  2     8 Glass, Philip              Glassworks                      Facades
  2     9 Glass, Philip              Glassworks                      Closing
  2    10 Glass, Philip              Songs from Liquid Days          Freezing
  2    11 Glass, Philip              Songs from Liquid Days          Lightning
 40     8 Rameau, Jean-Philippe      Music for A Baroque Festival    Concert 3 'Les Indes Galantes'
```

Finally, when we check for tracks in the Disc Number field with <u>integer and positive signed input data</u> (e.g. "3"), it searches for discs with the number "3" (not 33, or 13, or 31) through the database and successfully retrieves all data with Disc Number 3. Therefore, this field does not support the property of partial search.



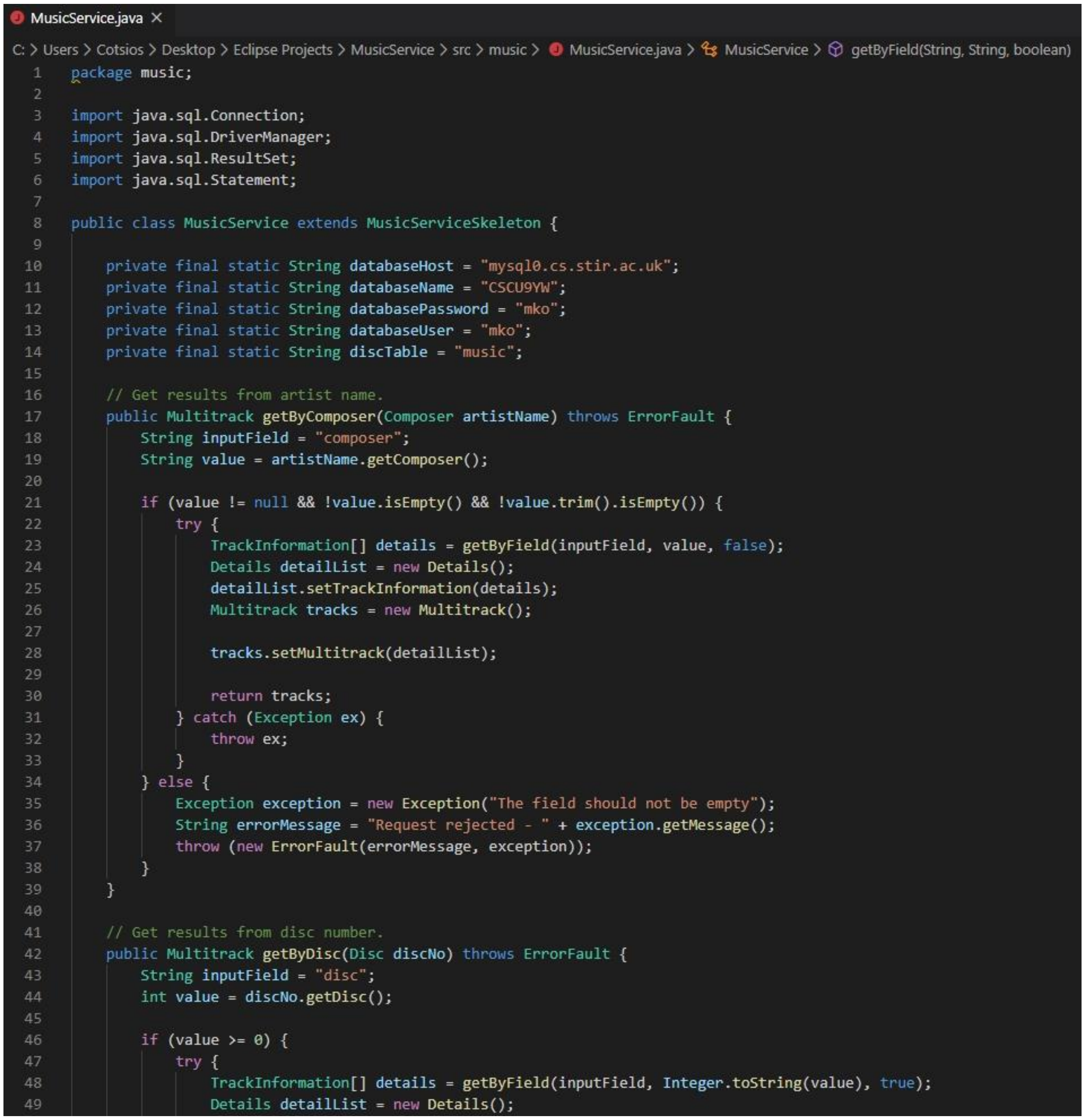
```
Disc Track Composer/Artist           Work                            Title
  3     1 Jarre, Jean-Michel         Chronologie                     Chronologie 1
  3     2 Jarre, Jean-Michel         Chronologie                     Chronologie 2
  3     3 Jarre, Jean-Michel         Chronologie                     Chronologie 3
  3     4 Jarre, Jean-Michel         Chronologie                     Chronologie 4
  3     5 Jarre, Jean-Michel         Chronologie                     Chronologie 5
  3     6 Jarre, Jean-Michel         Chronologie                     Chronologie 6
  3     7 Jarre, Jean-Michel         Chronologie                     Chronologie 7
  3     8 Jarre, Jean-Michel         Chronologie                     Chronologie 8
  3     9 Jarre, Jean-Michel         Magnetic Fields                 Magnetic Fields 1
  3    10 Jarre, Jean-Michel         Magnetic Fields                 Magnetic Fields 2
  3    11 Jarre, Jean-Michel         Magnetic Fields                 Magnetic Fields 3
  3    12 Jarre, Jean-Michel         Magnetic Fields                 Magnetic Fields 4
  3    13 Jarre, Jean-Michel         Magnetic Fields                 Magnetic Fields 5
```

## 3 Project completion

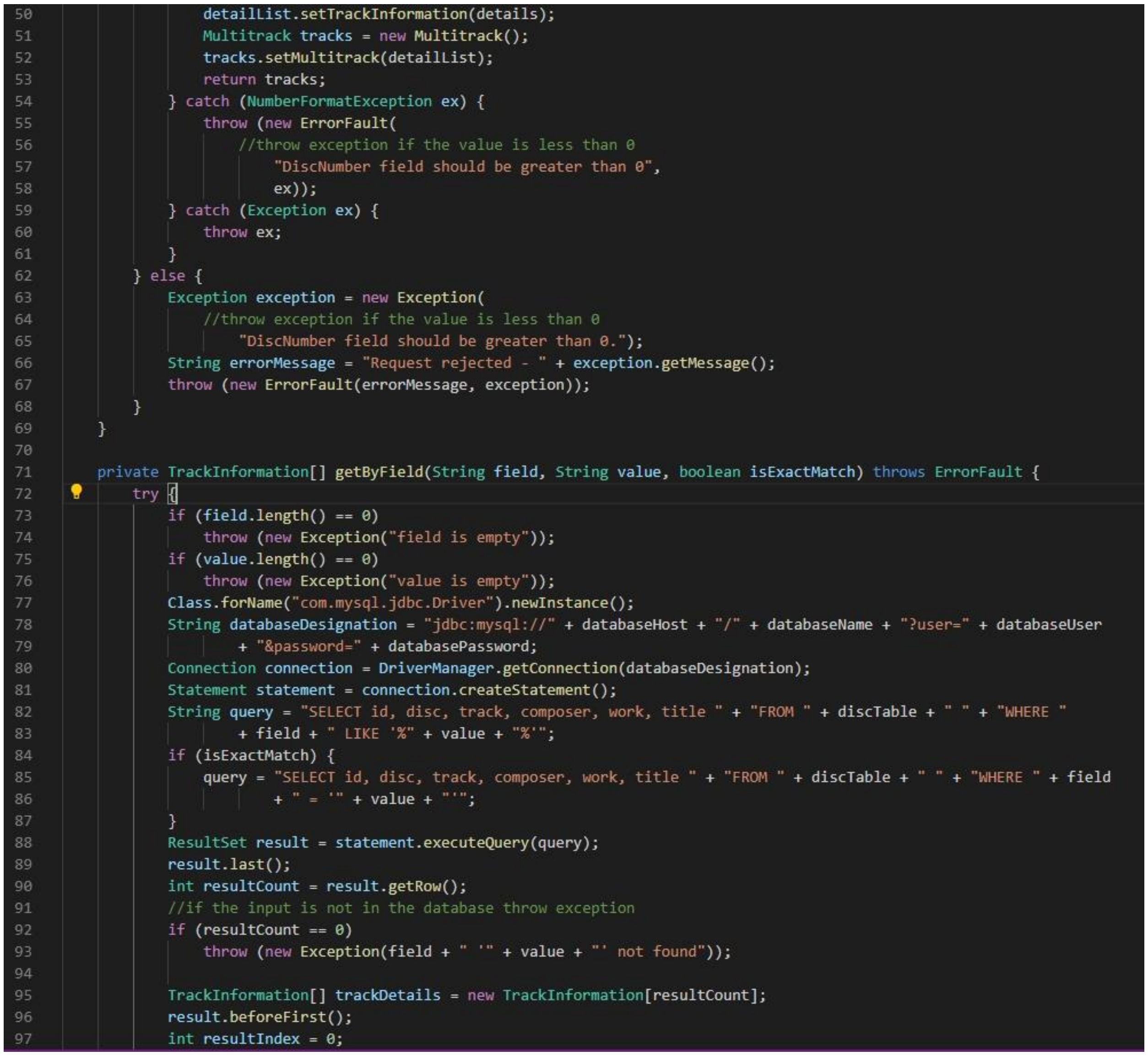
In order to complete the project assigned, we successful addressed and implemented all requirements. As shown in Section 2, the project is fully functional as all functionality from the assignment has been completed and added to the web service.

## 4 Appendix

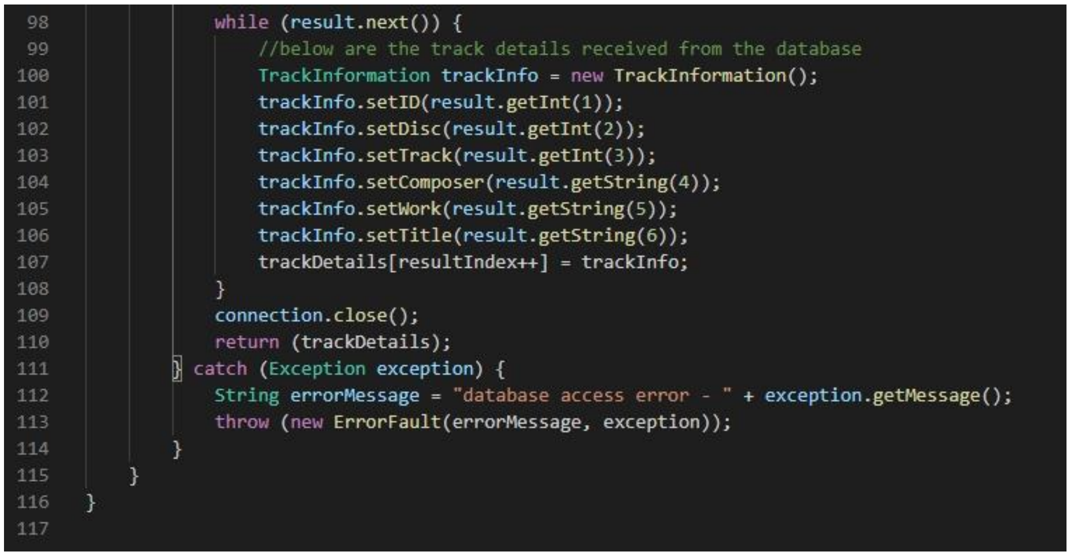
### 4.1 MusicService.java


MusicService.java ×
C: > Users > Cotsios > Desktop > Eclipse Projects > MusicService > src > music > MusicService.java > MusicService > getByField(String, String, boolean)

```java
package music;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class MusicService extends MusicServiceSkeleton {

    private final static String databaseHost = "mysql0.cs.stir.ac.uk";
    private final static String databaseName = "CSCU9YW";
    private final static String databasePassword = "mko";
    private final static String databaseUser = "mko";
    private final static String discTable = "music";

    // Get results from artist name.
    public Multitrack getByComposer(Composer artistName) throws ErrorFault {
        String inputField = "composer";
        String value = artistName.getComposer();

        if (value != null && !value.isEmpty() && !value.trim().isEmpty()) {
            try {
                TrackInformation[] details = getByField(inputField, value, false);
                Details detailList = new Details();
                detailList.setTrackInformation(details);
                Multitrack tracks = new Multitrack();

                tracks.setMultitrack(detailList);

                return tracks;
            } catch (Exception ex) {
                throw ex;
            }
        } else {
            Exception exception = new Exception("The field should not be empty");
            String errorMessage = "Request rejected - " + exception.getMessage();
            throw (new ErrorFault(errorMessage, exception));
        }
    }

    // Get results from disc number.
    public Multitrack getByDisc(Disc discNo) throws ErrorFault {
        String inputField = "disc";
        int value = discNo.getDisc();

        if (value >= 0) {
            try {
                TrackInformation[] details = getByField(inputField, Integer.toString(value), true);
                Details detailList = new Details();
```

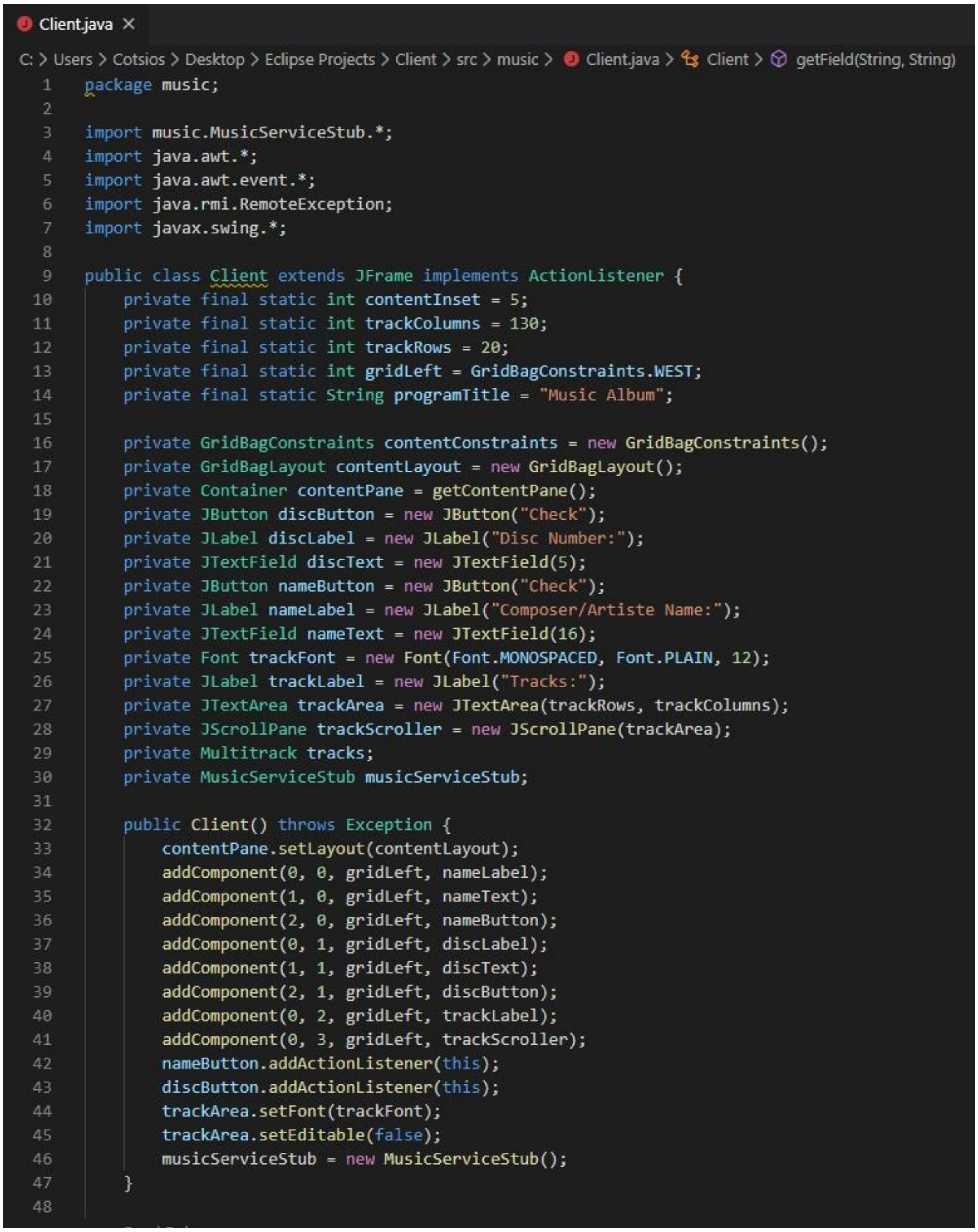


```
                detailList.setTrackInformation(details);
                Multitrack tracks = new Multitrack();
                tracks.setMultitrack(detailList);
                return tracks;
            } catch (NumberFormatException ex) {
                throw (new ErrorFault(
                    //throw exception if the value is less than 0
                        "DiscNumber field should be greater than 0",
                        ex));
            } catch (Exception ex) {
                throw ex;
            }
        } else {
            Exception exception = new Exception(
                //throw exception if the value is less than 0
                    "DiscNumber field should be greater than 0.");
            String errorMessage = "Request rejected - " + exception.getMessage();
            throw (new ErrorFault(errorMessage, exception));
        }
    }

    private TrackInformation[] getByField(String field, String value, boolean isExactMatch) throws ErrorFault {
        try {
            if (field.length() == 0)
                throw (new Exception("field is empty"));
            if (value.length() == 0)
                throw (new Exception("value is empty"));
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            String databaseDesignation = "jdbc:mysql://" + databaseHost + "/" + databaseName + "?user=" + databaseUser
                    + "&password=" + databasePassword;
            Connection connection = DriverManager.getConnection(databaseDesignation);
            Statement statement = connection.createStatement();
            String query = "SELECT id, disc, track, composer, work, title " + "FROM " + discTable + " " + "WHERE "
                    + field + " LIKE '%" + value + "%'";
            if (isExactMatch) {
                query = "SELECT id, disc, track, composer, work, title " + "FROM " + discTable + " " + "WHERE " + field
                        + " = '" + value + "'";
            }
            ResultSet result = statement.executeQuery(query);
            result.last();
            int resultCount = result.getRow();
            //if the input is not in the database throw exception
            if (resultCount == 0)
                throw (new Exception(field + " '" + value + "' not found"));

            TrackInformation[] trackDetails = new TrackInformation[resultCount];
            result.beforeFirst();
            int resultIndex = 0;
```



```
            while (result.next()) {
                //below are the track details received from the database
                TrackInformation trackInfo = new TrackInformation();
                trackInfo.setID(result.getInt(1));
                trackInfo.setDisc(result.getInt(2));
                trackInfo.setTrack(result.getInt(3));
                trackInfo.setComposer(result.getString(4));
                trackInfo.setWork(result.getString(5));
                trackInfo.setTitle(result.getString(6));
                trackDetails[resultIndex++] = trackInfo;
            }
            connection.close();
            return (trackDetails);
        } catch (Exception exception) {
            String errorMessage = "database access error - " + exception.getMessage();
            throw (new ErrorFault(errorMessage, exception));
        }
    }
}
```

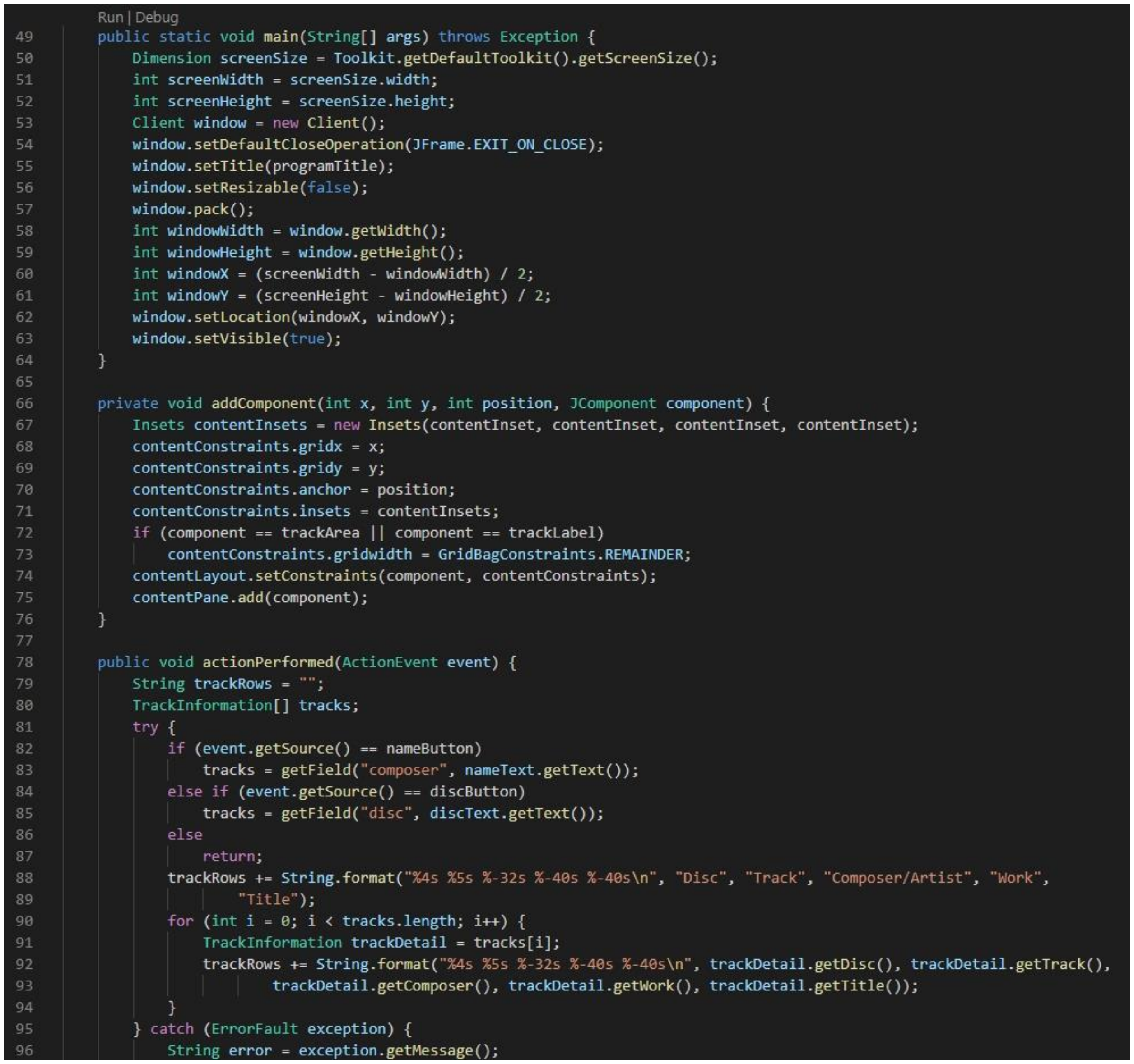
## 4.2 Client.java



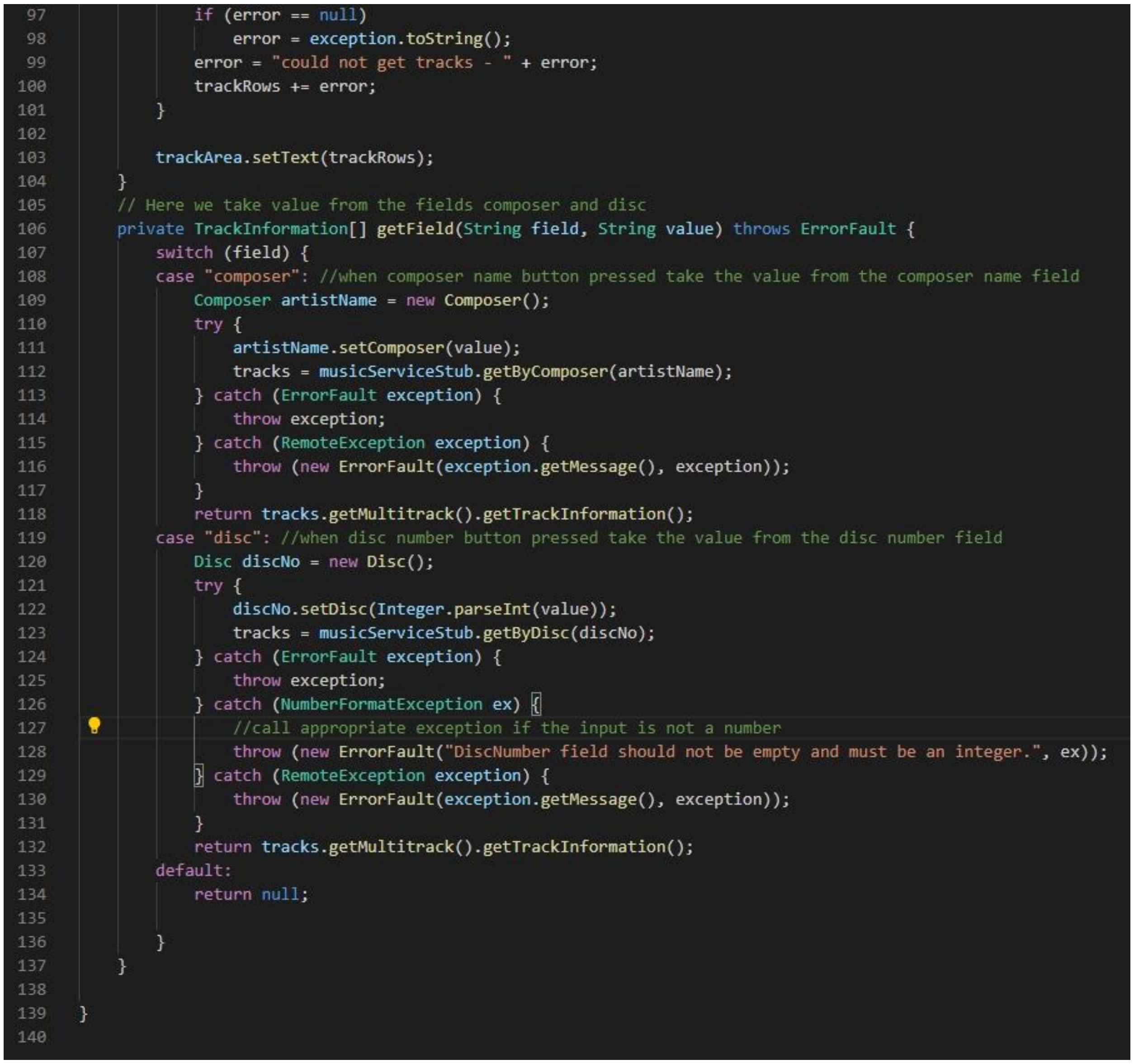
Client.java ×

C: > Users > Cotsios > Desktop > Eclipse Projects > Client > src > music > Client.java > Client > getField(String, String)

```java
package music;

import music.MusicServiceStub.*;
import java.awt.*;
import java.awt.event.*;
import java.rmi.RemoteException;
import javax.swing.*;

public class Client extends JFrame implements ActionListener {
    private final static int contentInset = 5;
    private final static int trackColumns = 130;
    private final static int trackRows = 20;
    private final static int gridLeft = GridBagConstraints.WEST;
    private final static String programTitle = "Music Album";

    private GridBagConstraints contentConstraints = new GridBagConstraints();
    private GridBagLayout contentLayout = new GridBagLayout();
    private Container contentPane = getContentPane();
    private JButton discButton = new JButton("Check");
    private JLabel discLabel = new JLabel("Disc Number:");
    private JTextField discText = new JTextField(5);
    private JButton nameButton = new JButton("Check");
    private JLabel nameLabel = new JLabel("Composer/Artiste Name:");
    private JTextField nameText = new JTextField(16);
    private Font trackFont = new Font(Font.MONOSPACED, Font.PLAIN, 12);
    private JLabel trackLabel = new JLabel("Tracks:");
    private JTextArea trackArea = new JTextArea(trackRows, trackColumns);
    private JScrollPane trackScroller = new JScrollPane(trackArea);
    private Multitrack tracks;
    private MusicServiceStub musicServiceStub;

    public Client() throws Exception {
        contentPane.setLayout(contentLayout);
        addComponent(0, 0, gridLeft, nameLabel);
        addComponent(1, 0, gridLeft, nameText);
        addComponent(2, 0, gridLeft, nameButton);
        addComponent(0, 1, gridLeft, discLabel);
        addComponent(1, 1, gridLeft, discText);
        addComponent(2, 1, gridLeft, discButton);
        addComponent(0, 2, gridLeft, trackLabel);
        addComponent(0, 3, gridLeft, trackScroller);
        nameButton.addActionListener(this);
        discButton.addActionListener(this);
        trackArea.setFont(trackFont);
        trackArea.setEditable(false);
        musicServiceStub = new MusicServiceStub();
    }

```

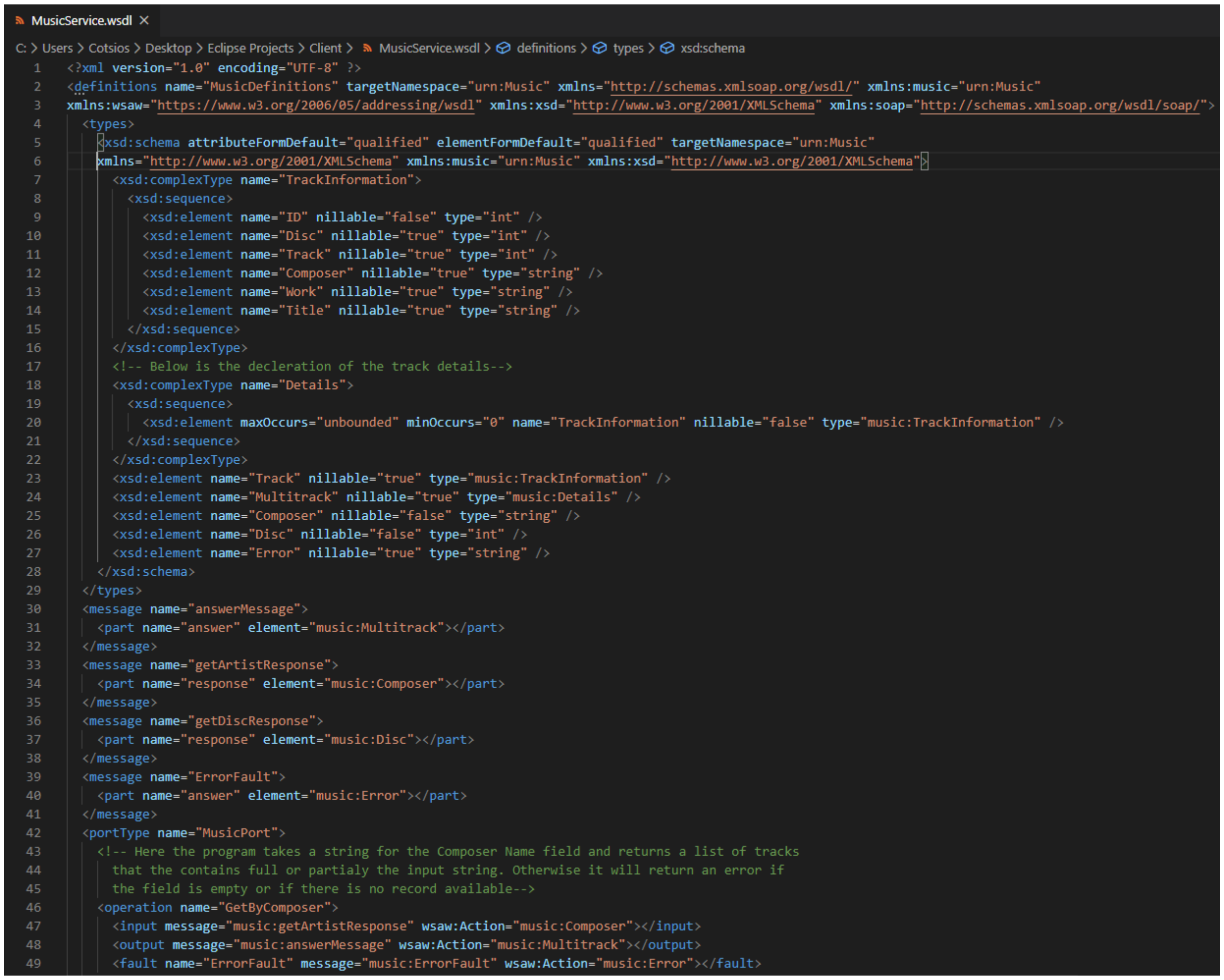


```
Run | Debug
public static void main(String[] args) throws Exception {
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    int screenWidth = screenSize.width;
    int screenHeight = screenSize.height;
    Client window = new Client();
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setTitle(programTitle);
    window.setResizable(false);
    window.pack();
    int windowWidth = window.getWidth();
    int windowHeight = window.getHeight();
    int windowX = (screenWidth - windowWidth) / 2;
    int windowY = (screenHeight - windowHeight) / 2;
    window.setLocation(windowX, windowY);
    window.setVisible(true);
}

private void addComponent(int x, int y, int position, JComponent component) {
    Insets contentInsets = new Insets(contentInset, contentInset, contentInset, contentInset);
    contentConstraints.gridx = x;
    contentConstraints.gridy = y;
    contentConstraints.anchor = position;
    contentConstraints.insets = contentInsets;
    if (component == trackArea || component == trackLabel)
        contentConstraints.gridwidth = GridBagConstraints.REMAINDER;
    contentLayout.setConstraints(component, contentConstraints);
    contentPane.add(component);
}

public void actionPerformed(ActionEvent event) {
    String trackRows = "";
    TrackInformation[] tracks;
    try {
        if (event.getSource() == nameButton)
            tracks = getField("composer", nameText.getText());
        else if (event.getSource() == discButton)
            tracks = getField("disc", discText.getText());
        else
            return;
        trackRows += String.format("%4s %5s %-32s %-40s %-40s\n", "Disc", "Track", "Composer/Artist", "Work",
                "Title");
        for (int i = 0; i < tracks.length; i++) {
            TrackInformation trackDetail = tracks[i];
            trackRows += String.format("%4s %5s %-32s %-40s %-40s\n", trackDetail.getDisc(), trackDetail.getTrack(),
                    trackDetail.getComposer(), trackDetail.getWork(), trackDetail.getTitle());
        }
    } catch (ErrorFault exception) {
        String error = exception.getMessage();
```

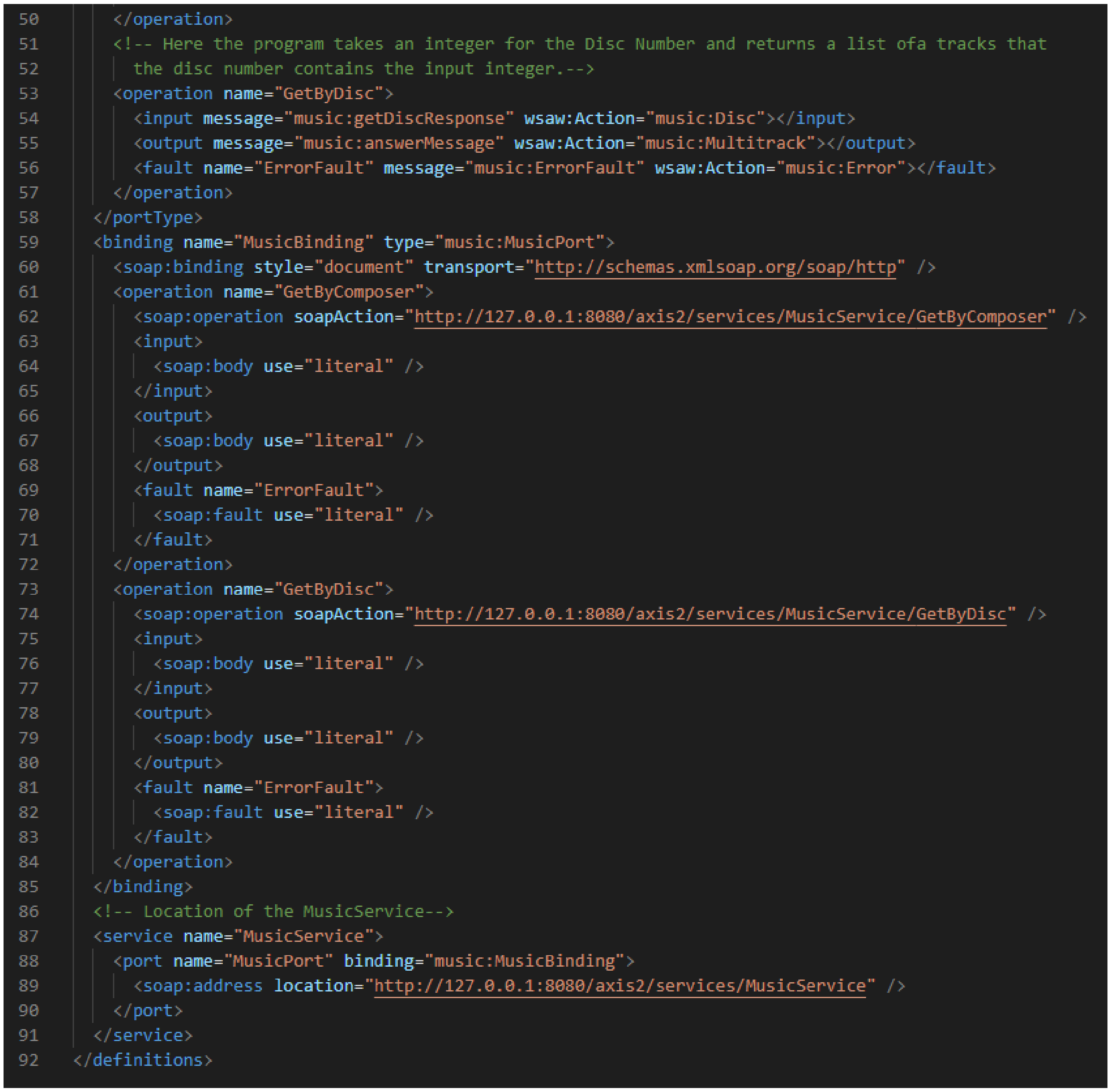


```
            if (error == null)
                error = exception.toString();
            error = "could not get tracks - " + error;
            trackRows += error;
        }

        trackArea.setText(trackRows);
    }
    // Here we take value from the fields composer and disc
    private TrackInformation[] getField(String field, String value) throws ErrorFault {
        switch (field) {
        case "composer": //when composer name button pressed take the value from the composer name field
            Composer artistName = new Composer();
            try {
                artistName.setComposer(value);
                tracks = musicServiceStub.getByComposer(artistName);
            } catch (ErrorFault exception) {
                throw exception;
            } catch (RemoteException exception) {
                throw (new ErrorFault(exception.getMessage(), exception));
            }
            return tracks.getMultitrack().getTrackInformation();
        case "disc": //when disc number button pressed take the value from the disc number field
            Disc discNo = new Disc();
            try {
                discNo.setDisc(Integer.parseInt(value));
                tracks = musicServiceStub.getByDisc(discNo);
            } catch (ErrorFault exception) {
                throw exception;
            } catch (NumberFormatException ex) {
                //call appropriate exception if the input is not a number
                throw (new ErrorFault("DiscNumber field should not be empty and must be an integer.", ex));
            } catch (RemoteException exception) {
                throw (new ErrorFault(exception.getMessage(), exception));
            }
            return tracks.getMultitrack().getTrackInformation();
        default:
            return null;

        }
    }

}
```

## 4.3 Services.wsdl


MusicService.wsdl
C: > Users > Cotsios > Desktop > Eclipse Projects > Client > MusicService.wsdl > definitions > types > xsd:schema

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="MusicDefinitions" targetNamespace="urn:Music" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:music="urn:Music"
xmlns:wsaw="https://www.w3.org/2006/05/addressing/wsdl" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="urn:Music"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:music="urn:Music" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="TrackInformation">
        <xsd:sequence>
          <xsd:element name="ID" nillable="false" type="int" />
          <xsd:element name="Disc" nillable="true" type="int" />
          <xsd:element name="Track" nillable="true" type="int" />
          <xsd:element name="Composer" nillable="true" type="string" />
          <xsd:element name="Work" nillable="true" type="string" />
          <xsd:element name="Title" nillable="true" type="string" />
        </xsd:sequence>
      </xsd:complexType>
      <!-- Below is the decleration of the track details-->
      <xsd:complexType name="Details">
        <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="0" name="TrackInformation" nillable="false" type="music:TrackInformation" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="Track" nillable="true" type="music:TrackInformation" />
      <xsd:element name="Multitrack" nillable="true" type="music:Details" />
      <xsd:element name="Composer" nillable="false" type="string" />
      <xsd:element name="Disc" nillable="false" type="int" />
      <xsd:element name="Error" nillable="true" type="string" />
    </xsd:schema>
  </types>
  <message name="answerMessage">
    <part name="answer" element="music:Multitrack"></part>
  </message>
  <message name="getArtistResponse">
    <part name="response" element="music:Composer"></part>
  </message>
  <message name="getDiscResponse">
    <part name="response" element="music:Disc"></part>
  </message>
  <message name="ErrorFault">
    <part name="answer" element="music:Error"></part>
  </message>
  <portType name="MusicPort">
    <!-- Here the program takes a string for the Composer Name field and returns a list of tracks
      that the contains full or partialy the input string. Otherwise it will return an error if
      the field is empty or if there is no record available-->
    <operation name="GetByComposer">
      <input message="music:getArtistResponse" wsaw:Action="music:Composer"></input>
      <output message="music:answerMessage" wsaw:Action="music:Multitrack"></output>
      <fault name="ErrorFault" message="music:ErrorFault" wsaw:Action="music:Error"></fault>
```



```
    </operation>
    <!-- Here the program takes an integer for the Disc Number and returns a list ofa tracks that
      the disc number contains the input integer.-->
    <operation name="GetByDisc">
      <input message="music:getDiscResponse" wsaw:Action="music:Disc"></input>
      <output message="music:answerMessage" wsaw:Action="music:Multitrack"></output>
      <fault name="ErrorFault" message="music:ErrorFault" wsaw:Action="music:Error"></fault>
    </operation>
  </portType>
  <binding name="MusicBinding" type="music:MusicPort">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="GetByComposer">
      <soap:operation soapAction="http://127.0.0.1:8080/axis2/services/MusicService/GetByComposer" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
      <fault name="ErrorFault">
        <soap:fault use="literal" />
      </fault>
    </operation>
    <operation name="GetByDisc">
      <soap:operation soapAction="http://127.0.0.1:8080/axis2/services/MusicService/GetByDisc" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
      <fault name="ErrorFault">
        <soap:fault use="literal" />
      </fault>
    </operation>
  </binding>
  <!-- Location of the MusicService-->
  <service name="MusicService">
    <port name="MusicPort" binding="music:MusicBinding">
      <soap:address location="http://127.0.0.1:8080/axis2/services/MusicService" />
    </port>
  </service>
</definitions>
```