My program code:

```
package clientStream;

import org.eclipse.swt.SWT;
import org.eclipse.swt.custom.StyledText;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;

import taxCalculator.TaxCalculator;
import taxRulesDb.TaxRule;
import taxRulesDb.TaxRulesDb;

//GUI for client stream of tax calculator
public class ClientDialog {

	protected Shell shlWelcomeClient;

	/**
	 * Launch the application.
	 *
	 * @param args
	 */

	private String clientId = TaxCalculator.userId;
	/**
	 * Open the window.
	 *
	 * @wbp.parser.entryPoint
	 */
	public void open() {
		Display display = Display.getDefault();
		createContents();
		shlWelcomeClient.open();
		shlWelcomeClient.layout();
		while (!shlWelcomeClient.isDisposed()) {
			if (!display.readAndDispatch()) {
				display.sleep();
			}
		}
	}

	/**
	 * Create contents of the window.
	 */
	protected void createContents() {
		shlWelcomeClient = new Shell();
		shlWelcomeClient.setSize(450, 450);
		shlWelcomeClient.setText("Welcome Client");

		StyledText styledText = new StyledText(shlWelcomeClient, SWT.BORDER);//
income text box
		styledText.setBounds(88, 50, 207, 89);

		Button btnNewButton = new Button(shlWelcomeClient, SWT.NONE);
		btnNewButton.addSelectionListener(new SelectionAdapter() {// update
income button
			@Override
			public void widgetSelected(SelectionEvent e) {
```

```
                        String s = styledText.getText();

                        System.out.println("incometext box is" + s);


                        TaxCalculator.incomeDB.updateClientIncome(clientId, 
styledText.getText());

                        System.out.println("incomeDb is");

                        TaxCalculator.incomeDB.printIncomeDB();


                }


            });

            btnNewButton.setBounds(317, 50, 80, 27);

            btnNewButton.setText("Update");


            Button btnCancel = new Button(shlWelcomeClient, SWT.NONE);

            btnCancel.addSelectionListener(new SelectionAdapter() {// update cancel 
income button

                @Override

                public void widgetSelected(SelectionEvent e) {


        styledText.setText(TaxCalculator.incomeDB.retrieveIncome(clientId).toString())
;

                }

            });

            btnCancel.setBounds(317, 111, 80, 27);

            btnCancel.setText("Cancel");


            Label lblIncome = new Label(shlWelcomeClient, SWT.NONE);

            lblIncome.setBounds(166, 27, 61, 17);

            lblIncome.setText("Income");


            StyledText styledText_1 = new StyledText(shlWelcomeClient, SWT.BORDER | 
SWT.WRAP);// tax rules box
```

```
		styledText_1.setBounds(88, 204, 207, 89);

	styledText_1.setText(TaxCalculator.taxRulesDb.getDefaultTaxRules().toString()

	.concat(TaxCalculator.taxRulesDb.getCustomTaxRules(clientId)));

		Label lblTaxRules = new Label(shlWelcomeClient, SWT.NONE);
		lblTaxRules.setBounds(166, 179, 61, 17);
		lblTaxRules.setText("Tax Rules");

		Button btnUpdate = new Button(shlWelcomeClient, SWT.NONE);
		btnUpdate.addSelectionListener(new SelectionAdapter() {
			@Override
			public void widgetSelected(SelectionEvent e) {
				TaxCalculator.taxRulesDb.updateCustomTaxRules(clientId,
new TaxRule(styledText_1.getText()));
			}
		});
		btnUpdate.setBounds(317, 210, 80, 27);
		btnUpdate.setText("Update");

		Button btnCancel_1 = new Button(shlWelcomeClient, SWT.NONE);
		btnCancel_1.setBounds(317, 266, 80, 27);
		btnCancel_1.setText("Cancel");

		Text text = new Text(shlWelcomeClient, SWT.BORDER);
		text.setBounds(157, 332, 138, 27);

		Button btnCalculateTax = new Button(shlWelcomeClient, SWT.NONE);
		btnCalculateTax.addSelectionListener(new SelectionAdapter() {
			@Override
			public void widgetSelected(SelectionEvent e) {
        text.setText(Integer.toString(TaxRulesDb.getDefualtTaxRules()

        .apply(TaxCalculator.incomeDB.retrieveIncome(TaxCalculator.userId))));


                    }

                });

                btnCalculateTax.setBounds(317, 332, 80, 27);

                btnCalculateTax.setText("Calculate Tax");


                Label lblYourTax = new Label(shlWelcomeClient, SWT.NONE);

                lblYourTax.setBounds(88, 335, 61, 17);

                lblYourTax.setText("Your Tax:");


                Button btnExit = new Button(shlWelcomeClient, SWT.NONE);

                btnExit.addSelectionListener(new SelectionAdapter() {

                    @Override

                    public void widgetSelected(SelectionEvent e) {

                        shlWelcomeClient.close();


                    }

                });

                btnExit.setBounds(157, 384, 80, 27);

                btnExit.setText("Exit");


        }


}
```

```
package clientStream;
//Class for each client's income
public class ClientIncome {
            private String userId;
            private Income income;

            ClientIncome(String u, Income i){
```

```
			userId=u;
			income=i;
		}

		ClientIncome(String u, String i){
			userId=u;
			income = new Income(i);

		}


		public String getUserId() {
			return userId;
		}

		public void setUserId(String userId) {
			this.userId = userId;
		}

		public Income getIncome() {
			return income;
		}

		public void setIncome(Income income) {
			this.income = income;
		}

		@Override
		public String toString() {
			return userId+";"+income.toString();
		}

	}

package clientStream;


public class Income {
	private int employmentIncome;
	private int selfEmploymentIncome;
	private int capitalGains;
	//Client's income converted from the text box input
	public Income(String incomeString){
		//incomeString is assumed to be in the format of
		//"#; #; #"
		int first = incomeString.indexOf(';');
		int second= incomeString.indexOf(';', first+1);
		employmentIncome = Integer.parseInt(incomeString.substring(0, first));
		selfEmploymentIncome = Integer.parseInt(incomeString.substring(first+1, 
second));
		capitalGains = Integer.parseInt(incomeString.substring(second+1));
	}

	public int getEmploymentIncome() {
		return employmentIncome;
	}

	public void setEmploymentIncome(int employmentIncome) {
		this.employmentIncome = employmentIncome;
	}

	public int getSelfEmploymentIncome() {
		return selfEmploymentIncome;
	}

	public void setSelfEmploymentIncome(int selfEmploymentIncome) {
		this.selfEmploymentIncome = selfEmploymentIncome;
	}

	public int getCapitalGains() {
		return capitalGains;
	}

	public void setCapitalGains(int capitalGains) {
		this.capitalGains = capitalGains;
	}

	@Override
	public String toString() {//converted to be in the format of "#; #; #"
		return Integer.toString(employmentIncome).concat(";").concat

	(Integer.toString(selfEmploymentIncome)).concat(";").concat
				(Integer.toString(capitalGains));

	}

}

package clientStream;

import java.util.ArrayList;
//Database for income
public class IncomeDB {

	static private ArrayList<ClientIncome> clientIncomes = new ArrayList<>();

	// may convert to a file named "ClientIncomesDB.txt" or a DB later

	private void addClientIncome(ClientIncome e) {
		clientIncomes.add(e);
	}

	private void addClientIncome(String userId, Income income) {
		addClientIncome(new ClientIncome(userId, income));
	}

	public void addClientIncome(String userId, String incomeString) {
		System.out.println("Before adding income for " + userId + incomeString);
		Income i = new Income(incomeString);
		System.out.println(i.toString());
		addClientIncome(userId, i);
	}

	public Income retrieveIncome(String userId) {
		for (int i = 0; i < clientIncomes.size(); i++) {
			if (clientIncomes.get(i).getUserId().equalsIgnoreCase(userId))
				return clientIncomes.get(i).getIncome();
		}
		return null;
	}


	private void updateClientIncome(String userId, Income income) {
		for (int i = 0; i < clientIncomes.size(); i++) {
			if (clientIncomes.get(i).getUserId().equalsIgnoreCase(userId)) {
				clientIncomes.get(i).setIncome(income);
				return;
			}
		}
		// if new user, append into DB
		addClientIncome(userId, income);
	}

	public void updateClientIncome(String userId, String incomeString) {
		Income i = new Income(incomeString);
		updateClientIncome(userId, i);
	}

	public void printIncomeDB() {
		clientIncomes.forEach(client -> {
			System.out.println(client.toString());
		});

	}

}
package loginRegisterStream;
//Setting up the administrator account
public class AdminAccount {
		static private String adminName = new String("Admin"); //case
insensitive
		static private String adminPassWord = new String("Admin"); //case
sensitive
		//Checks for whether user is admin
		public boolean isAdmin(String name, String pwd) {
			if (adminName.equalsIgnoreCase(name) &&
					adminPassWord.equals(pwd))
				return true;
			else
				return false;
		}
	}
package loginRegisterStream;
```

```java
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;

import taxCalculator.TaxCalculator;

public class LogInDialog {


	protected Shell shlWelcomeToTax;
	private Text txtUserName;
	private Text txtPassWord;


	static private int failedTrial=0;
	protected static final int MAXTRIALS = 3; //max 3 fails to dispose the shell

	/**
	 * Launch the application.
	 * @param args
	 */
	/**
	 * Open the window.
	 * @wbp.parser.entryPoint
	 */
	public void open() {
		Display display = Display.getDefault();
		createContents();
		shlWelcomeToTax.open();
		shlWelcomeToTax.layout();
		while (!shlWelcomeToTax.isDisposed()) {
			if (!display.readAndDispatch()) {
				display.sleep();
			}
		}
	}



	/**
	 * Create contents of the window.
	 */
	protected void createContents() {
		shlWelcomeToTax = new Shell();
		shlWelcomeToTax.setSize(450, 300);
		shlWelcomeToTax.setText("Welcome to Tax Calculator");

		txtUserName = new Text(shlWelcomeToTax, SWT.BORDER);
		txtUserName.setBounds(186, 38, 130, 23);

		txtPassWord = new Text(shlWelcomeToTax, SWT.BORDER | SWT.PASSWORD);
		txtPassWord.setBounds(186, 88, 130, 23);

		Button btnNewButton = new Button(shlWelcomeToTax, SWT.NONE);
		btnNewButton.addSelectionListener(new SelectionAdapter() {
package loginRegisterStream;
```

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Dialog;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;
//GUI for registering account
public class RegisterDialog extends Dialog {

	protected Object result;
	protected Shell shlRegister;
	private Text text;
	private Text text_1;
	private UserAccountsDB userAccountsDB=new UserAccountsDB();

	/**
	 * Create the dialog.
	 * @param parent
	 * @param style
	 */
	public RegisterDialog(Shell parent, int style) {
		super(parent, style);
		setText("Register");
	}

	/**
	 * Open the dialog.
	 * @return the result
	 */
	public Object open() {
		createContents();
		shlRegister.open();
		shlRegister.layout();
		Display display = getParent().getDisplay();
		while (!shlRegister.isDisposed()) {
			if (!display.readAndDispatch()) {
				display.sleep();
			}
		}
		return result;
	}

	/**
	 * Create contents of the dialog.
	 */
	private void createContents() {
		shlRegister = new Shell(getParent(), SWT.DIALOG_TRIM |
SWT.PRIMARY_MODAL);
		shlRegister.setSize(450, 300);
		shlRegister.setText("Register");

		text = new Text(shlRegister, SWT.BORDER);
		text.setBounds(182, 62, 104, 23);

		text_1 = new Text(shlRegister, SWT.BORDER | SWT.PASSWORD);
		text_1.setBounds(182, 123, 104, 23);

		Label lblUser = new Label(shlRegister, SWT.NONE);
		lblUser.setBounds(103, 65, 61, 17);
		lblUser.setText("User");
		Label lblPassword = new Label(shlRegister, SWT.NONE);
		lblPassword.setBounds(103, 126, 61, 17);
		lblPassword.setText("Password");

		Button btnOk = new Button(shlRegister, SWT.NONE);
		btnOk.addSelectionListener(new SelectionAdapter() {
			@Override
			public void widgetSelected(SelectionEvent e) {
				System.out.println("NewUser: "+text.getText()+ "with pwd: 
" + text_1.getText());
				//Add user accounts
				userAccountsDB.addUserAccount(new 
String(text.getText()),new String(text_1.getText()));
				text.setText("");
				text_1.setText("");
				shlRegister.dispose();
			}
		});
		btnOk.setBounds(79, 208, 80, 27);
		btnOk.setText("OK");
		//Cancel button
		Button btnCancel = new Button(shlRegister, SWT.NONE);
		btnCancel.addSelectionListener(new SelectionAdapter() {
			@Override
			public void widgetSelected(SelectionEvent e) {
				shlRegister.dispose();
			}
		});
		btnCancel.setBounds(228, 208, 80, 27);
		btnCancel.setText("Cancel");
```

```
	}


}
package loginRegisterStream;
//possible values of StreamContext
public enum StreamContext {
	NOT_DEFINED, ADMIN_STREAM, CLIENT_STREAM;

}
package loginRegisterStream;

public class StreamContextTest {

	StreamContext streamContext;

	public StreamContextTest(StreamContext cntxt)
	{
		this.streamContext=cntxt;
	}

	public void printContext() {
		switch (streamContext){
	case NOT_DEFINED:
		System.out.println("Context is undefined.");
		break;
	case ADMIN_STREAM:
		System.out.println("Context is admin.");
		break;
	case CLIENT_STREAM:
		System.out.println("Context is client. ");
		break;
}
	}
}


			@Override

			public void widgetSelected(SelectionEvent e) {

				System.out.println("Log In button pressed");

				System.out.println("Input--user "+txtUserName.getText()+ "
Password "+ txtPassWord.getText());

				if
(TaxCalculator.adminAccount.isAdmin(txtUserName.getText(), txtPassWord.getText())) {

					System.out.println("Is Admin");


	TaxCalculator.streamContext=StreamContext.ADMIN_STREAM;
```

```
            //                  System.out.println("Set TaxCalculator.streamContext
to StreamContext.ADMIN_STREAM");
                                shlWelcomeToTax.close();
                          }
                          else {
                                String userId = TaxCalculator.userAccountsDB.isUser
(new String(txtUserName.getText()),
                                            new String(txtPassWord.getText()));
                                System.out.println("Got userId: "+userId);

                                if (userId != null) {
                                System.out.println("Is client with uuid: "+ userId);

      TaxCalculator.streamContext=StreamContext.CLIENT_STREAM;
                                TaxCalculator.userId=userId;
                                shlWelcomeToTax.close();
                                }else {
                                      System.out.println("Is not a client ");

      TaxCalculator.streamContext=StreamContext.NOT_DEFINED;
                                      failedTrial++;
                                      System.out.println("Failed "+failedTrial);
                                      if (failedTrial >= MAXTRIALS)
                                            shlWelcomeToTax.close();



                                }



                    };
                    txtUserName.setText("");
                    txtPassWord.setText("");
            }});
```

```
		btnNewButton.setBounds(100, 189, 80, 27);

		btnNewButton.setText("Log In");


		Button btnNewButton_1 = new Button(shlWelcomeToTax, SWT.NONE);

		btnNewButton_1.addSelectionListener(new SelectionAdapter() {

			@Override

			public void widgetSelected(SelectionEvent e) {

				System.out.println("Register button pressed");

				RegisterDialog registerDialog = new 
RegisterDialog(shlWelcomeToTax, 0);

				registerDialog.open();

			}

		});

		btnNewButton_1.setBounds(257, 189, 80, 27);

		btnNewButton_1.setText("Register");


		Label lblUsername = new Label(shlWelcomeToTax, SWT.NONE);

		lblUsername.setBounds(119, 41, 61, 17);

		lblUsername.setText("UserName");


		Label lblPassword = new Label(shlWelcomeToTax, SWT.NONE);

		lblPassword.setBounds(119, 91, 61, 17);

		lblPassword.setText("PassWord");

	}

}
package loginRegisterStream;
//Class for user accounts
public class UserAccount {
	private String userName;
	private String passWord;
	private String userId;

	public UserAccount(String userName, String passWord, String userId) {
		super();
		this.userName = userName;
		this.passWord = passWord;
		this.userId = userId;
	}


	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 String getUserId() {
		return userId;
	}
	public void setUserId(String userId) {
		this.userId = userId;
	}



}

package loginRegisterStream;


import java.util.ArrayList;

import java.util.UUID;


public class UserAccountsDB {

	//Stores user accounts

	static private ArrayList<UserAccount> usersAccounts = new ArrayList<>();


	// may convert to a file named "userAccountsDB.txt" or a DB later


	public void addUserAccount(String name, String pwd) {

		usersAccounts.add(new UserAccount(name, pwd,
UUID.randomUUID().toString()));

	}
	@SuppressWarnings("static-access")
	public String isUser(String name, String pwd) {
		//if name and pwd match one user, return unser ID;
		//otherwise null;
		for (int i = 0; i < UserAccountsDB.usersAccounts.size(); i++) {

			if (usersAccounts.get(i).getUserName().equalsIgnoreCase(name)
					&& usersAccounts.get(i).getPassWord().equals(pwd))
				return usersAccounts.get(i).getUserId();
		}
		return null;
	}

	public void printUserAcountsDB() {
		usersAccounts.forEach(user -> {
			System.out.println(
					"Name: " + user.getUserName() + " PassWord: " +
user.getPassWord() + " UUID: " + user.getUserId());
		});

	}

}
package taxCalculator;

import clientStream.ClientDialog;
import clientStream.IncomeDB;
import loginRegisterStream.AdminAccount;
import loginRegisterStream.LogInDialog;
import loginRegisterStream.StreamContext;
import loginRegisterStream.UserAccountsDB;
import taxRulesDb.TaxRulesDb;
//main controller of the whole program
public class TaxCalculator {

	public static UserAccountsDB userAccountsDB = new UserAccountsDB();
	public static IncomeDB incomeDB = new IncomeDB();
	public static TaxRulesDb taxRulesDb = new TaxRulesDb();
	public static AdminAccount adminAccount = new AdminAccount();
	public static StreamContext streamContext = StreamContext.NOT_DEFINED;
	public static String userId = null;


	public static void main(String[] args) {

		try {
			LogInDialog logInDialog = new LogInDialog();
			logInDialog.open();
		} catch (Exception e) {
			//e.printStackTrace();
		};
		//Identifies which stream the user should be in
		switch (streamContext) {
		case NOT_DEFINED:
			System.out.println("Context is undefined.");
			break;
		case ADMIN_STREAM:
			System.out.println("Context is admin.");
			break;
		case CLIENT_STREAM:
			System.out.println("Context is client with userID " + userId);
			try {
```

```
                          TaxRulesDb.updateDefaultTaxRules("0% under 30,000 for
taxable income; 15% under 80,000; 30% under 100000; 50% above 100,000; "
                                       + "capital gains count as half for taxable
income");
                          // for testing purpose, default tax rules is managed in
adminStream.
                          System.out.println("TaxRulesDb is: " +
taxRulesDb.toString());



                          ClientDialog window = new ClientDialog();
                          window.open();
                    } catch (Exception e) {
                          //e.printStackTrace();
                    }
                    break;
             }
      }
}
package taxRulesDb;

public class CustomTaxRules extends TaxRules {
      private String userId;

      public CustomTaxRules(String userId2, TaxRule aRule) {
            // a user with his first rule
            userId=userId2;
            super.update(aRule);
      }

      public String getUserId() {
            return userId;
      }

      public void setUserId(String userId) {
            this.userId = userId;
      }

      @Override
      public String toString() {
            return userId.concat("\n").concat(super.toString());
      }
```

```
	public boolean matchUserId(String userId2) {
		// TODO Auto-generated method stub
		return userId.equalsIgnoreCase(userId2);
	}

	public void addNewUser(String userId2, TaxRule aRule) {
		userId = userId2;
		super.update(aRule);
	}

}
```

```
package taxRulesDb;

import clientStream.Income;
//Single tax rule
public class TaxRule {
	private String readableRule;
	private String backendRule;

	public TaxRule(String r) {// only readable rules
		readableRule = r;
		backendRule = r;
	}
	//already translated rules
	public TaxRule(String r, String b) {
		readableRule = r;
		backendRule = b;
	}

	public String getReadableRule() {
		return readableRule;
	}

	public void setReadableRule(String readableRule) {
		this.readableRule = readableRule;
	}

	public String getBackendRule() {
		return backendRule;
	}

	public void setBackendRule(String backendRule) {
		this.backendRule = backendRule;
	}
	//checking if the mentioned rule is this rule for purposes such as updating or
retrieving
	public boolean readableRuleEquals(TaxRule aTaxRule) {
		if (this.readableRule.equalsIgnoreCase(aTaxRule.readableRule))
			return true;
		else
			return false;

	}
	@Override
	public String toString() {
		return readableRule;

	}
	//Current default tax rule (Current Year Canadian Tax Rule)
	public int apply(Income aIncome) {
		// a stub, to be developed;
		// Only applies the defaultTaxRule of
		// "0% under 30,000; 15% under 80,000; 30% under 100000; 50% above
100,000"
		int sum = aIncome.getEmploymentIncome() +
aIncome.getSelfEmploymentIncome() + aIncome.getCapitalGains() / 2;
		if (sum <= 30000)
			return 0;
		if (sum <= 80000)
			return (int) ((sum - 30000) * 0.15);
		if (sum <= 100000)
			return (int) ((80000-30000)*0.15+ (sum - 80000) * 0.30);

		return (int) ((80000-30000)*0.15+ (100000 - 80000) * 0.30 + (sum -
100000) * 0.5);
	}

}

package taxRulesDb;


import java.util.ArrayList;


import clientStream.Income;

//Many tax rules

public class TaxRules {

	private ArrayList<TaxRule> taxRules = new ArrayList<>();




	public void update(TaxRule aTaxRule) {

		//if one rule's readable rule match, change its backend rule;

		//if no match, add the rule


		for (int i=0; i<taxRules.size(); i++) {

			if (taxRules.get(i).readableRuleEquals(aTaxRule)) {

				taxRules.remove(i);
```

```
                taxRules.add(aTaxRule);
                return;
            }
        }

        taxRules.add(aTaxRule);
    }




    @Override
    public String toString() {
        String s="";
        for (int i=0; i<taxRules.size(); i++) {
            s=s.concat(taxRules.get(i).toString());
            s=s.concat ("\n");
        }
        return s;
    }


    public TaxRules() {
        // TODO Auto-generated constructor stub
    }



    public int apply(Income income) {
        return taxRules.get(0).apply(income); // a stub, to be developed
```

```
	}
}

package taxRulesDb;

import java.util.ArrayList;

public class TaxRulesDb {
	static private TaxRules defaultTaxRules = new TaxRules();
	static private ArrayList<CustomTaxRules> customTaxRules = new ArrayList<>();
	// one set of defaultTaxRules, many sets of custom rules for many clients

	public static ArrayList<CustomTaxRules> getCustomTaxRules() {
		return customTaxRules;
	}

	public static void updateDefaultTaxRules(String s) {
		defaultTaxRules.update(new TaxRule(s));
	}

	public TaxRules getDefaultTaxRules() {
		return defaultTaxRules;
	}

	public void updateCustomTaxRules(String userId, TaxRule aRule) {
		// if user exists, update his rule;
		// otherwise, add the new user with his rule;
		for (int i = 0; i < customTaxRules.size(); i++) {
			if (customTaxRules.get(i).matchUserId(userId)) {
				customTaxRules.get(i).update(aRule);
				return;
			}
		}
		TaxRulesDb.customTaxRules.add(new CustomTaxRules(userId, aRule));

	}

	@Override
	public String toString() {
		String s = "";

		for (int i = 0; i < customTaxRules.size(); i++) {
			s = s.concat(customTaxRules.get(i).toString());
			s = s.concat("\n");
		}

		return defaultTaxRules.toString().concat(s);
	}

	public String getCustomTaxRules(String clientId) {
		// return the rules for particular client
		for (int i = 0; i < customTaxRules.size(); i++) {
			if (customTaxRules.get(i).matchUserId(null))
				return customTaxRules.get(i).toString();
		}
		return "";
	}

	public static TaxRules getDefualtTaxRules() {
		return defaultTaxRules;
	}

}
```