```
unit frmMain;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
  DB, ADODB,
  LCRConfig, LCRGlobals, CostingSteps, LCRCostVars;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    edtExcelInput: TEdit;
    btnExcelInput: TButton;
    OpenDialog1: TOpenDialog;
    btnMakeSample: TButton;
    Label2: TLabel;
    edtSampleName: TEdit;
    Label3: TLabel;
    edtMinPerCategory: TEdit;
    Memo1: TMemo;
    ckbxNoReplication: TCheckBox;
    ckbxMakeBaseline: TCheckBox;
    ckbxMakeOption: TCheckBox;
    Label4: TLabel;
    Label5: TLabel;
    edtVariableDatabase: TEdit;
    edtCostLogicWorkbook: TEdit;
    btnVariableDatabase: TButton;
    btnCostWorkbook: TButton;
    Label6: TLabel;
    cmbOptions: TComboBox;
    Label7: TLabel;
    edtSmallProxyPop: TEdit;
    Label8: TLabel;
    Label9: TLabel;
    edtPWS90_BP1: TEdit;
    edtPWS90_BP2: TEdit;
    chkProxyRecords: TCheckBox;
    CheckBox1: TCheckBox;
    Edit1: TEdit;
    Button1: TButton;
    edtBaselineVar: TEdit;
    Label10: TLabel;
    Button2: TButton;
    Label11: TLabel;
    cmbxLSLLevel: TComboBox;
    chkUseSavedBins: TCheckBox;
    procedure btnExcelInputClick(Sender: TObject);
    procedure btnMakeSampleClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnVariableDatabaseClick(Sender: TObject);
    procedure btnCostWorkbookClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
    Config: TLCRConfig;
    BaseCostVars: TCostVars;
    BaseCostSteps: TCostingSteps;
    ScenCostVars: TCostVars;
    ScenCostSteps: TCostingSteps;

    procedure OpenCostVars(aConfig: TLCRConfig);
    procedure WriteLog(BaselineSampleFilename, OptionSampleFilename: string);
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

uses PWSReplicator;

procedure TForm1.btnCostWorkbookClick(Sender: TObject);
begin
  if OpenDialog1.Execute then
    edtCostLogicWorkbook.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnExcelInputClick(Sender: TObject);
begin
  if OpenDialog1.Execute then
    edtExcelInput.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnMakeSampleClick(Sender: TObject);
var
  SDWIS: TPWSReplicator;
  i, j, k: integer;
```

```
begin
  RandSeed := 1;
  UserSeeds := true;

  SDWIS := TPWSReplicator.Create;
  Config := TLCRConfig.Create;
  Config.ConnectExcelEPProbs;
  Config.BaseVarData:=edtBaselineVar.Text;

  // read Baseline and NDWAC databases
  // create Baseline and NDWAC TCostingSteps objects
  OpenCostVars(Config);

  SDWIS.SDWISFile := edtExcelInput.Text;
  SDWIS.OptionName := cmbOptions.Text;
  SDWIS.SampleName := edtSampleName.Text;
  SDWIS.MinPerCategory := StrToInt(edtMinPerCategory.Text);
  SDWIS.MakeProfileSample:=Checkbox1.Checked;
  SDWIS.SampRate:=strtofloat(Edit1.Text);
  SDWIS.UseSavedBins := chkUseSavedBins.Checked;
  SDWIS.LSLLevel := cmbxLSLLevel.Text;

  if chkProxyRecords.Checked then
    SDWIS.ProxyRecords := true
  else
    SDWIS.ProxyRecords := false;

  SDWIS.SmallProxyPop := StrToInt(edtSmallProxyPop.Text);
  SDWIS.SmallProxyLabel := edtSmallProxyPop.Text;
  SDWIS.PWS90PctBp1 := StrToInt(edtPWS90_BP1.Text);
  SDWIS.PWS90PctBp2 := StrToInt(edtPWS90_BP2.Text);

  if ckbxNoReplication.Checked then
    SDWIS.NoReplication := true
  else
    SDWIS.NoReplication := false;

  SDWIS.Config := Config;

  SDWIS.BaseCostSteps := BaseCostSteps;

  // read original sample xlsx file, count number PWS in each category
  // create category weights
  SDWIS.ReadSDWIS;

  for k := 1 to 2 do
    for i := 1 to 9 do
      for j := 1 to 2 do
```

```
         Memo1.Lines.Add(k.ToString + ' ' + i.ToString + ' ' + j.ToString + ' ' +
                         SDWIS.CategoryCount[k,i,j].ToString + ' ' +
SDWIS.CategoryRep[k,i,j].ToString + ' ' +
                         SDWIS.CategoryWeight[k,i,j].ToString);

  RG.fSC:=0;
  RG.fBC:=0;
  // write Baseline sample
  if ckbxMakeBaseline.Checked then begin
    SDWIS.WriteBaseline3;
    //CSL('SRnd Base:'+RG.fSC.ToString);
    //CSL('BRnd Base:'+RG.fBC.ToString);
  end;

  // write option sample
  // this file is based on the Baseline sample file
  if ckbxMakeOption.Checked then
  begin
    SDWIS.ScenCostSteps := ScenCostSteps;
    SDWIS.WriteOption;
  end;

  WriteLog(SDWIS.BaselineSampleFilename, SDWIS.OptionSampleFilename);

  BaseCostSteps.Free;
  BaseCostVars.Free;
  ScenCostSteps.Free;
  ScenCostVars.Free;
  Config.Free;
  SDWIS.Free;

  ShowMessage('Output files written');
end;

procedure TForm1.btnVariableDatabaseClick(Sender: TObject);
begin
  if OpenDialog1.Execute then
    edtVariableDatabase.Text := OpenDialog1.FileName;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
    edtBaselineVar.Text := OpenDialog1.FileName;
end;

procedure TForm1.Button2Click(Sender: TObject);
var s1,s2 : string;
```

```
    F1,F2,O1,O2 : TextFile;
    cnt,errcnt,i,j,c1,c2 : integer;
    founddiff : boolean;
    F1Var,F2Var,F1Val,F2Val :  TStringList;
    VarMiss : TStringLIst;
begin
  if not OpenDialog1.Execute() then exit;
  s1:=OpenDialog1.FileName;
  if not OpenDialog1.Execute() then exit;
  s2:=OpenDialog1.FileName;
  assignfile(F1,s1);
  reset(F1);
  assignfile(F2,s2);
  reset(F2);

  c1:=0; c2:=0;
  while not eof(F1) do begin
    inc(c1);
    readln(F1,s1);
  end;
  reset(F1);
  while not eof(F2) do begin
    inc(c2);
    readln(F2,s1);
  end;
  reset(F2);
  if c1<>c2 then begin
    closefile(F1);
    closefile(F2);
    exit;
  end;

  F1Var:=TStringList.Create;
  F1Var.StrictDelimiter:=True;
  F2Var:=TStringList.Create;
  F2Var.StrictDelimiter:=True;
  F1Val:=TStringList.Create;
  F1Val.StrictDelimiter:=True;
  F2Val:=TStringList.Create;
  F2Val.StrictDelimiter:=True;

  readln(F1,s1);
  F1Var.CommaText:=S1;
  readln(F2,s2);
  F2Var.CommaText:=S2;
  for i:=0 to F1Var.Count-1 do begin
    j:=F2Var.IndexOf(F1Var[i]);
    if j<0 then begin
    end;
  end;
  for i:=0 to F2Var.Count-1 do begin
    j:=F1Var.IndexOf(F2Var[i]);
    if j<0 then begin
    end;
  end;


  cnt:=0; errcnt:=0;
  VarMiss:=TstringList.Create;
  VarMiss.Sorted:=True;
  while not eof(F1) do begin
    inc(cnt);
    readln(F1,s1);
    readln(F2,s2);
    F1Val.CommaText:=S1;
    F2Val.CommaText:=S2;
    for i:=0 to F1Var.Count-1 do begin
      j:=i;
      if (F1Var[i]<>F2Var[j]) then
      if j>-1 then begin

        if F1Val[i]<>F2Val[j] then begin
          inc(errcnt);

          if VarMiss.IndexOf(F1Var[i])<0 then VarMiss.Add(F1Var[i]);
        end;

      end;
    end;
    if errcnt>2000 then break;
  end;

  closefile(F1);
  closefile(F2);
  F1Var.Free;
  F2Var.Free;
  F1Val.Free;
  F2Val.Free;
  showmessage('Done');
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  AppPath:=ExtractFilePath(Application.ExeName);
  DataPath:=AppPath+'data\';
  UserPath:=AppPath+'user\';
  HelpPath:=AppPath+'help\';
  csl('code logging');
end;

procedure TForm1.OpenCostVars(aConfig: TLCRConfig);
var
  ADOCon: TADOConnection;
  qDesc, qData: TADOQuery;
  CostLogicWorkbook, VariableDatabase: string;
begin
  CostLogicWorkbook := edtCostLogicWorkbook.Text;
  VariableDatabase := edtVariableDatabase.Text;

  ADOCon:=TADOCOnnection.Create(nil);
  ADOCon.ConnectionString:=format(ADOConStr,[aConfig.BaseVarData]);
  qDesc:=TADOQuery.Create(nil);
  qDesc.Connection:=ADOCon;
  qData:=TADOQuery.Create(nil);
  qData.Connection:=ADOCon;
  qDesc.SQL.Add('select * from InputDesc');
  qData.SQL.Add('select * from InputValues');


  qDesc.Open;
  qData.Open;

BaseCostVars:=TCostVars.Create(qDesc,qData,ChangeFileExt(aConfig.BaseVarData,'.xlsm'
));
  qData.Close;
  qDesc.Close;
  AdoCon.Close;

  ADOCon.ConnectionString:=format(ADOConStr,[VariableDatabase]);
  qData.Open;
  qDesc.Open;

ScenCostVars:=TCostVars.Create(qDesc,qData,ChangeFileExt(VariableDatabase,'.xlsm'),
BaseCostVars);
  AdoCon.Close;
  qData.Free;
  qDesc.Free;
  ADOCon.Free;

  BaseCostSteps := TCostingSteps.Create(BaseCostVars, aConfig.BaseCostSteps, '', 0,
0, True);
  ScenCostSteps := TCostingSteps.Create(ScenCostVars, CostLogicWorkbook, '', 0, 0,
False);
end;
```

```
procedure TForm1.WriteLog(BaselineSampleFilename, OptionSampleFilename: string);
var
  today: TDateTime;
  filename: string;
  SLLog: TStringList;
begin
  today := Now;
  filename := datapath + 'sample_run_' + FormatDateTime('mmddyyyy',today) + '_' +
FormatDateTime('hhnn',today) + '.log';
  SLLog := TStringList.Create;
  SLLog.Add('Run date: ' + FormatDateTime('mm/dd/yyyy',today) + ' ' +
FormatDateTime('hh:nn',today));
  SLLog.Add('Option name: ' + cmbOptions.text);
  SLLog.Add('Sample name: ' + edtSampleName.text);
  SLLog.Add('Excel input: ' + edtExcelInput.text);
  SLLog.Add('Baseline variable database: ' + edtBaselineVar.text);
  SLLog.Add('Option variable database: ' + edtVariableDatabase.text);
  SLLog.Add('Option costing logic workbook: ' + edtCostLogicWorkbook.text);
  SLLog.Add('Minimum # PWS: ' + edtMinPerCategory.text);
  if ckbxNoReplication.Checked then
    SLLog.Add('Do not replicate: Checked')
  else
    SLLog.Add('Do not replicate: Not checked');
  if ckbxMakeBaseline.Checked then
    SLLog.Add('Make baseline: Checked')
  else
    SLLog.Add('Make baseline: Not checked');
  if ckbxMakeOption.Checked then
    SLLog.Add('Make option: Checked')
  else
    SLLog.Add('Make option: Not checked');
  if chkProxyRecords.Checked then
    SLLog.Add('Create proxy records: Checked')
  else
    SLLog.Add('Create proxy records: Not checked');
  if chkUseSavedBins.Checked then
    SLLog.Add('Use saved bins: Checked')
  else
    SLLog.Add('Use saved bins: Not checked');
  SLLog.Add('Small proxy pop cutoff: ' + edtSmallProxyPop.text);
  SLLog.Add('LSL level: ' + cmbxLSLLevel.text);
  SLLog.Add('PWS90Pct Bp1: ' + edtPWS90_BP1.text);
  SLLog.Add('PWS90Pct Bp2: ' + edtPWS90_BP2.text);
  if CheckBox1.Checked then
    SLLog.Add('Create profile sample: Checked')
  else
    SLLog.Add('Create profile sample: Not checked');
  SLLog.Add('Profile sample: ' + Edit1.text);

  if BaselineSampleFilename <> '' then
  begin
    SLLog.Add('Baseline sample filename: ' + BaselineSampleFilename);
    SLLog.Add('Baseline sample file datetime: ' +
DateTimeToStr(FileDateToDateTime(FileAge(BaselineSampleFilename))));
  end;

  if OptionSampleFilename <> '' then
  begin
    SLLog.Add('Option sample filename: ' + OptionSampleFilename);
    SLLog.Add('Option sample file datetime: ' +
DateTimeToStr(FileDateToDateTime(FileAge(OptionSampleFilename))));
  end;

  SLLog.SaveToFile(filename);

  SLLog.Free;
end;

end.
```