```
unit frmMain;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, UITypes, Vcl.ExtCtrls,
DBClient,
  CodeSiteLogging,LCRBenefits,
  LCRModel, LCRConfig, LCRGlobals, LCRMetricCollector, LCRCostVars, CostingSteps,
LCRPreCompile,
  LCRCompiledCost;

type
  TForm1 = class(TForm)
    btnRunModel: TButton;
    btnStopModel: TButton;
    OpenDialog1: TOpenDialog;
    Timer1: TTimer;
    Label1: TLabel;
    Label2: TLabel;
    edtRunName: TEdit;
    Label3: TLabel;
    mmDescription: TMemo;
    Label4: TLabel;
    edtCostLogicWorkbook: TEdit;
    btnOpenCostLogicWrkbk: TButton;
    Label5: TLabel;
    edtVariableDatabase: TEdit;
    btnOpenVariableDatabase: TButton;
    Label6: TLabel;
    Panel1: TPanel;
    rbCWS: TRadioButton;
    rbNTNCWS: TRadioButton;
    Label7: TLabel;
    Panel2: TPanel;
    rb3Pct: TRadioButton;
    rb7Pct: TRadioButton;
    Panel3: TPanel;
    rbMeanRun: TRadioButton;
    rbVariableRun: TRadioButton;
    edtIterations: TEdit;
    cmbOptions: TComboBox;
    labIterations: TLabel;
    Panel4: TPanel;
    rbFullRun: TRadioButton;
    rbBaselineOnly: TRadioButton;
```

```
rbOptionOnly: TRadioButton;
CheckBox1: TCheckBox;
Label8: TLabel;
Label9: TLabel;
edtBaselineSDWISSample: TEdit;
btnOpenBaselineSample: TButton;
edtOptionSDWISSample: TEdit;
btnOpenOptionSample: TButton;
chkLeadBins: TCheckBox;
cbDebug: TCheckBox;
cbCCTToFull: TCheckBox;
Button1: TButton;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
edtSmallProxyPop: TEdit;
edtPWS90_BP1: TEdit;
edtPWS90_BP2: TEdit;
Label13: TLabel;
cmbSchoolOption: TComboBox;
rbBoth: TRadioButton;
chkSmallSysFlex: TCheckBox;
cbNoBLAvg: TCheckBox;
cmbCCTCostEq: TComboBox;
Label14: TLabel;
edIQVS: TEdit;
Label15: TLabel;
Label16: TLabel;
edIQDR: TEdit;
edCVDR: TEdit;
Label17: TLabel;
Label18: TLabel;
edChildBL: TEdit;
Label19: TLabel;
CheckBox2: TCheckBox;
Label20: TLabel;
edtVolProg: TEdit;
lbStat: TLabel;
CheckBox3: TCheckBox;
Button2: TButton;
Label21: TLabel;
CheckBox4: TCheckBox;
Label22: TLabel;
Edit1: TEdit;
cbBenAll: TCheckBox;
Label23: TLabel;
cbBenByYear: TCheckBox;
procedure btnRunModelClick(Sender: TObject);
```

```
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure rbMeanRunClick(Sender: TObject);
    procedure rbVariableRunClick(Sender: TObject);
    procedure btnOpenCostLogicWrkbkClick(Sender: TObject);
    procedure btnOpenVariableDatabaseClick(Sender: TObject);
    procedure btnOpenBaselineSampleClick(Sender: TObject);
    procedure btnOpenOptionSampleClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure btnStopModelClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
    RunName: string;
    OptionName: string;
    InitShow, InitActive, Stop: boolean;
    HiddenForm: boolean;
    CCTCostEquationLevel: string;

    procedure SaveLCRRunData(Filename: string);
    procedure SetParameters(Filename: string);
    procedure Status(Msg : string; var aStop : boolean);
  public
    { Public declarations }
    CurConfig: TLCRConfig;
    ModelRunning: boolean;
    LCRModel: TLCRModel;
    CurRunOutput : string;

    procedure ModelIsDone(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

uses DB, inifiles, SafewaterUncertBucket, frmTestParser;

{$R *.dfm}

procedure TForm1.btnOpenBaselineSampleClick(Sender: TObject);
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'CSV File|*.csv';
  OpenDialog1.FileName := '';
```

```
  if OpenDialog1.Execute then
    edtBaselineSDWISSample.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenCostLogicWrkbkClick(Sender: TObject);
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Excel Workbook|*.xlsx';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtCostLogicWorkbook.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenOptionSampleClick(Sender: TObject);
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'CSV File|*.csv';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtOptionSDWISSample.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenVariableDatabaseClick(Sender: TObject);
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Access database|*.accdb';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtVariableDatabase.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnRunModelClick(Sender: TObject);
var
  S,FN,FNP : string;
begin
  if cmbOptions.ItemIndex = -1 then
    exit;

  if MessageDlg('Run with current settings?',mtConfirmation,[mbYes,mbNo],0) <> mrYes
then exit;

  S:='';

  if edtRunName.Text <> 'Run Name' then
    OpenDialog1.FileName := edtRunName.Text;

  OpenDialog1.InitialDir := UserPath;
  if not OpenDialog1.Execute then exit;
```

```
  S:=OpenDialog1.FileName;
  FN:=ChangeFileExt(ExtractFileName(S),'');

  edtRunName.Text := FN;
  Application.ProcessMessages;
  OptionName := cmbOptions.Items[cmbOptions.ItemIndex];
  CCTCostEquationLevel := cmbCCTCostEq.Items[cmbCCTCostEq.ItemIndex];

  NoRandom:=Checkbox1.Checked;
  if NoRandom then RandSeed:=1;

  CurConfig.RunName:=FN;
  CurConfig.OptionName := OptionName;
  CurConfig.RunDescription := StringReplace(mmDescription.Lines.Text, #$D#$A, ' ',
[rfReplaceAll]);
  CurConfig.BasePWSDataFile := edtBaselineSDWISSample.Text;
  CurConfig.ScenPWSDataFile := edtOptionSDWISSample.Text;
  CurConfig.ScenCostSteps := edtCostLogicWorkbook.Text;
  CurConfig.ScenVarData := edtVariableDatabase.Text;
  CurConfig.Debug:=cbDebug.Checked;
  CurConfig.CCTToFull:=cbCCTToFull.Checked;
  CurConfig.CCTCostEquationLevel := CCTCostEquationLevel;
  CurConfig.BenefitsAll := cbBenAll.Checked;

  CurConfig.IQValueSens:=  strtoint(edIQVS.Text);
  CurConfig.IQDRSens:=  strtoint(edIQDR.Text);
  CurConfig.CVDDRSens:=strtoint(edCVDR.Text);
  CurConfig.ChildBLSens:=strtoint(edChildBL.Text);
  CurConfig.VolLeadProg := StrToInt(edtVolProg.Text);
  CurConfig.BenByYear := cbBenByYear.Checked;

  if rbCWS.Checked then
  begin
    CurConfig.SystemType := sysCWS;
    CurConfig.RunSysType := 'CWS';
  end
  else
  if rbNTNCWS.Checked then
  begin
    CurConfig.SystemType := sysNTNC;
    CurConfig.RunSysType := 'NTNCWS';
  end
  else
  if rbBoth.Checked then
  begin
    CurConfig.SystemType := sysCWS;
    CurConfig.RunSysType := 'Both';
  end;
```

```
if rb3Pct.Checked then
  CurConfig.DiscountRate := 0.03
else
if rb7Pct.Checked then
  CurConfig.DiscountRate := 0.07;

if rbMeanRun.Checked then
begin
  CurConfig.MeanUncertLevel := true;
  CurConfig.VariabilityRun := false;
end
else
if rbVariableRun.Checked then
begin
  CurConfig.VariabilityRun := true;
  CurConfig.MeanUncertLevel := false;
  CurConfig.NumberOfVLoops := strToInt(edtIterations.Text);
end;

CurConfig.RunBaselineOnly := rbBaselineOnly.Checked;
CurConfig.RunOptionOnly := rbOptionOnly.Checked;
CurConfig.RunDifference := rbFullRun.Checked;
CurConfig.RunNoBLAveraging := cbNoBLAvg.Checked;
CurConfig.OutputLeadBins := chkLeadBins.Checked;
CurConfig.SmallSystemFlexibility := chkSmallSysFlex.Checked;

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

CurConfig.SchoolOption := cmbSchoolOption.Items[cmbSchoolOption.ItemIndex];
CurConfig.UseCompiledCost := CheckBox4.Checked;

RunName:=CurConfig.RunName;
FNP:=ExtractFilePath(OpenDialog1.FileName);
FN:=ChangeFileExt(FN,'.swc');

_UseCompiled := CurConfig.UseCompiledCost;
CurConfig.Save(FNP+FN);
ModelRunning:=True;

LCRModel:=TLCRModel.Create(FNP+FN);
LCRModel.OnProgress := status;
LCRModel.MicroOutput := CheckBox3.Checked;
CurRunOutput:=ChangeFileExt(FNP+FN,'.swo');

LCRModel.OnModelDone:=ModelIsDone;
```

```
  Screen.Cursor:=crHourGlass;

  LCRModel.Initialize(nil,FNP);

  Screen.Cursor:=crDefault;
  btnRunModel.Enabled:=False;
  btnStopModel.Enabled:=True;

  LCRModel.Run;
  showmessage('done');
end;

procedure TForm1.btnStopModelClick(Sender: TObject);
begin
  Stop:=True;
end;

procedure TForm1.Button1Click(Sender: TObject);
var B : TLCRBenefits;
begin
  B:=TLCRBenefits.create(CurConfig, nil, nil, 'a');
  B.CalcBinMove(1,12,1,1000,False,btMandatory,True);
  B.Free;
end;

procedure TForm1.Button2Click(Sender: TObject);
var T : TLCRPreCompile;
    var BFile, SFIle : string;
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter:='Excel (*.xlsx) |*.xlsx';
  OpenDialog1.Title := 'Select Baseline Workbook';
  if OpenDialog1.Execute then begin
    BFile := OpenDialog1.FileName;
    OpenDialog1.Title := 'Select Option Workbook';
    if OpenDialog1.Execute then begin
      sFile := OpenDialog1.FileName;
      T:=TLCRPreCompile.create(edit1.text,sFile, bFIle);
      T.Go;
      T.Free;
      showmessage('done');
      Close;
    end;
  end;
  OpenDialog1.Title := '';
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
```

```
begin
  CodeSite.Enabled:=CheckBox2.Checked;
  CSL('Codesight enabled');
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  if InitActive then
  begin
    CodeSite.Enabled:=False;
    Label21.Caption := 'Compiled Costs Base: '+_CworkBookBaseline+'
'+_CWorkBookDate;
    Label22.Caption := 'Compiled Costs Option: '+_CworkBookOption;

    if MessageDlg('Do you want to load a saved configuration?  Hit Yes to load or No
to use the default settings.',mtConfirmation,[mbYes,mbNo],0)=mrYes then
    begin
      OpenDialog1.InitialDir := UserPath;
      OpenDialog1.Filter:='SW Config (*.swc) |*.swc';
      if OpenDialog1.Execute then begin
        CurConfig.Load(OpenDialog1.FileName);
        cmbOptions.ItemIndex := cmbOptions.Items.IndexOf(CurConfig.OptionName);
        cmbCCTCostEq.ItemIndex :=
cmbCCTCostEq.Items.IndexOf(CurConfig.CCTCostEquationLevel);
        cmbSchoolOption.ItemIndex :=
cmbSchoolOption.Items.IndexOf(CurConfig.SchoolOption);
      end;
    end;

    edtRunName.Text := CurConfig.RunName;
    mmDescription.Lines.Text := CurConfig.RunDescription;
    edtBaselineSDWISSample.Text := CurConfig.BasePWSDataFile;
    edtOptionSDWISSample.Text := CurConfig.ScenPWSDataFile;
    edtCostLogicWorkbook.Text := CurConfig.ScenCostSteps;
    edtVariableDatabase.Text := CurConfig.ScenVarData;
    edIQVS.Text:=inttostr(CurConfig.IQValueSens);
    edIQDR.Text:=inttostr(CurConfig.IQDRSens);
    edCVDR.Text:=inttostr(CurConfig.CVDDRSens);
    edChildBL.Text:=inttostr(CurConfig.ChildBLSens);
    edtVolProg.Text := IntToStr(CurConfig.VolLeadProg);
    edtPWS90_BP1.Text:=inttostr(CurConfig.PWS90PctBp1);
    edtPWS90_BP2.Text:=inttostr(CurConfig.PWS90PctBp2);
    cbBenAll.Checked := CurConfig.BenefitsAll;
    cbBenByYear.Checked := CurConfig.BenByYear;

    if CurConfig.RunSysType = 'CWS' then
      rbCWS.Checked := True
    else
```

```
    if CurConfig.RunSysType = 'NTNCWS' then
      rbNTNCWS.Checked := True
    else
    if CurConfig.RunSysType = 'Both' then
      rbBoth.Checked := True;

    if Round(CurConfig.DiscountRate*100)/100 = 0.03 then
      rb3Pct.Checked := true
    else
    if Round(CurConfig.DiscountRate*100)/100 = 0.07 then
      rb7Pct.Checked := true;

    rbBaselineOnly.Checked := CurConfig.RunBaselineOnly;
    rbOptionOnly.Checked := CurConfig.RunOptionOnly;
    rbFullRun.Checked := CurConfig.RunDifference;
    chkLeadBins.Checked := CurConfig.OutputLeadBins;
    chkSmallSysFlex.Checked := CurConfig.SmallSystemFlexibility;
    cbNoBLAvg.Checked:=CurConfig.RunNoBLAveraging;
    edtSmallProxyPop.Text := CurConfig.SmallProxyPop.ToString;

    InitActive := false;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  InitShow := true;
  InitActive := true;
  edtIterations.Enabled := false;

  AppPath := ExtractFilePath(Application.ExeName);
  DataPath := AppPath + 'data\';
  UserPath := AppPath + 'user\';
  HelpPath := AppPath + 'help\';

  CurConfig := TLCRConfig.Create;

  ModelRunning := false;

  HiddenForm := false;
  if ParamStr(1) <> '' then
  begin
    Application.ShowMainForm := false;
    HiddenForm := true;
    SetParameters(ParamStr(1));
  end;
end;
```

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  CurConfig.Free;
end;

procedure TForm1.ModelIsDone(Sender: TObject);
begin
  Timer1.Enabled:=False;

  LCRModel.Destroy;
  ModelRunning:=False;

  if not HiddenForm then
  begin
    btnRunModel.Enabled:=True;
    btnStopModel.Enabled:=False;
  end;

  SaveLCRRunData(CurRunOutput);

  if HiddenForm then
    Application.Terminate;
end;

procedure TForm1.rbMeanRunClick(Sender: TObject);
begin
  edtIterations.Enabled := false;
end;

procedure TForm1.rbVariableRunClick(Sender: TObject);
begin
  edtIterations.Enabled := true;
end;

procedure TForm1.SaveLCRRunData(Filename: string);
var
  T: TCategoryOutputReader;
  cdsOutputData: TClientDataset;
begin
  cdsOutputData := TClientDataSet.Create(nil);
  cdsOutputData.Close;

  cdsOutputData.FieldDefs.Clear;
  with cdsOutputData do
  begin
    with FieldDefs.AddFieldDef do
    begin
      Name := 'Category';
```

```
  DataType := ftString;
  Size := 100;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'MetricClass';
  DataType := ftString;
  Size := 15;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Metric';
  DataType := ftString;
  Size := 80;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Option';
  DataType := ftString;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'DiscountRate';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Statistic';
  DataType := ftString;
  Size := 10;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'P1';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'P5';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Mean';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
```

```
      Name := 'P95';
      DataType := ftFloat;
    end;
    with FieldDefs.AddFieldDef do
    begin
      Name := 'P99';
      DataType := ftFloat;
    end;
    with FieldDefs.AddFieldDef do
    begin
      Name := 'sDR';
      DataType := ftString;
    end;

    with IndexDefs do
    begin
      Clear;
      AddIndexDef.Name := 'USEIDX';
      AddIndexDef.Fields := 'Category;Metric;Option;DiscountRate;Statistic';
      AddIndexDef.Options := [];
    end;

  end;

  cdsOutputData.CreateDataSet;
  cdsOutputData.LogChanges := false;

  T := TCategoryOutputReader.Create(ChangeFileExt(Filename, '.swomn'));

  T.DumpToDataset(cdsOutputData, OptionName);
  T.DumpToTabDelim(ChangeFileExt(Filename, '.tab'), true, CurConfig.NoICRCosts);

  T.Free;
  cdsOutputData.Free;
end;

procedure TForm1.SetParameters(Filename: string);
var
  iniFile: TIniFile;
  RunName, OptionName, CCTCostEquationLevel: string;
  BaselineSDWIS, OptionSDWIS, OptionCostingLogic, OptionVariables: string;
  Population, DiscountRate, RunType: string;
  FNP: string;
begin
  iniFile := TIniFile.Create(AppPath + '\' + Filename);

  RunName := iniFile.ReadString('Params','RunName',RunName);
  OptionName := iniFile.ReadString('Params','OptionName',OptionName);
```

```
  BaselineSDWIS := iniFile.ReadString('Params','BaselineSDWIS',BaselineSDWIS);
  OptionSDWIS := iniFile.ReadString('Params','OptionSDWIS',OptionSDWIS);
  OptionCostingLogic :=
iniFile.ReadString('Params','OptionCostingLogic',OptionCostingLogic);
  OptionVariables := iniFile.ReadString('Params','OptionVariables',OptionVariables);
  Population := iniFile.ReadString('Params','Population',Population);
  DiscountRate := iniFile.ReadString('Params','DiscountRate',DiscountRate);
  RunType := iniFile.ReadString('Params','RunType',RunType);
  CCTCostEquationLevel :=
iniFile.ReadString('Params','CCTCostEquationLevel',CCTCostEquationLevel);

  iniFile.Free;

  CurConfig.RunName:=RunName;
  CurConfig.OptionName := OptionName;
  CurConfig.BasePWSDataFile := BaselineSDWIS;
  CurConfig.ScenPWSDataFile := OptionSDWIS;
  CurConfig.ScenCostSteps := OptionCostingLogic;
  CurConfig.ScenVarData := OptionVariables;
  CurConfig.CCTCostEquationLevel := CCTCostEquationLevel;

  if Population = 'CWS' then
    CurConfig.SystemType := sysCWS
  else
  if Population = 'NTNCWS' then
    CurConfig.SystemType := sysNTNC;

  if DiscountRate = '3%' then
    CurConfig.DiscountRate := 0.03
  else
  if DiscountRate = '7%' then
    CurConfig.DiscountRate := 0.07;

  CurConfig.RunBaselineOnly := false;
  CurConfig.RunOptionOnly := false;
  CurConfig.RunDifference := false;

  if RunType = 'Baseline' then
    CurConfig.RunBaselineOnly := true
  else if RunType = 'Option' then
    CurConfig.RunOptionOnly := true
  else if RunType = 'Incremental' then
    CurConfig.RunDifference := true;

  FNP:=UserPath;
  RunName:=UserPath + ChangeFileExt(RunName,'.swc');

  CurConfig.Save(RunName);
```

```
  ModelRunning:=True;

  LCRModel:=TLCRModel.Create(RunName);
  LCRModel.OnProgress := status;
  LCRModel.MicroOutput := CheckBox3.Checked;
  CurRunOutput:=ChangeFileExt(RunName,'.swo');

  LCRModel.OnModelDone:=ModelIsDone;

  LCRModel.Initialize(nil,FNP);
  Stop:=false;

  LCRModel.Run;
end;

procedure TForm1.Status(Msg: string; var aStop: boolean);
begin
  aStop:=Stop;
  lbStat.Caption := Msg;
  Application.ProcessMessages;
end;

end.
```