Table of contents

Prerequisites:

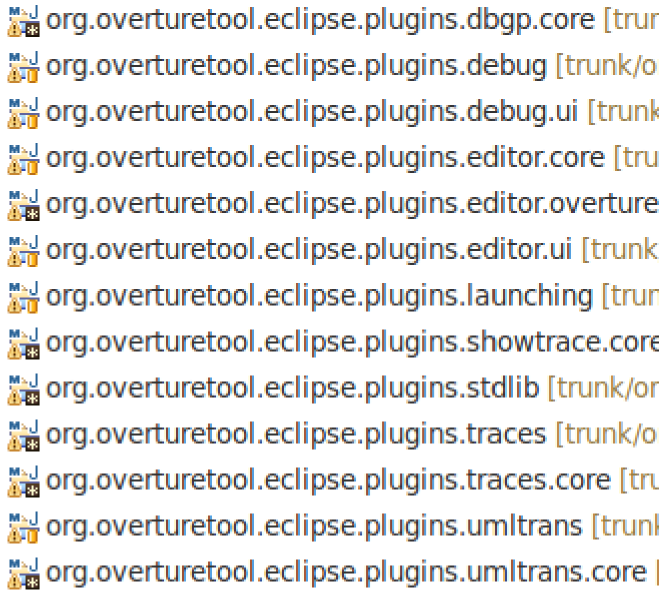
To be able to make these exercises a few things is needed:

- An Eclipse IDE with the Plug-in Development Environment (PDE)
- Checked all of the Overture project out using the guide Kenneth made
- Installed the plug-in DLTK

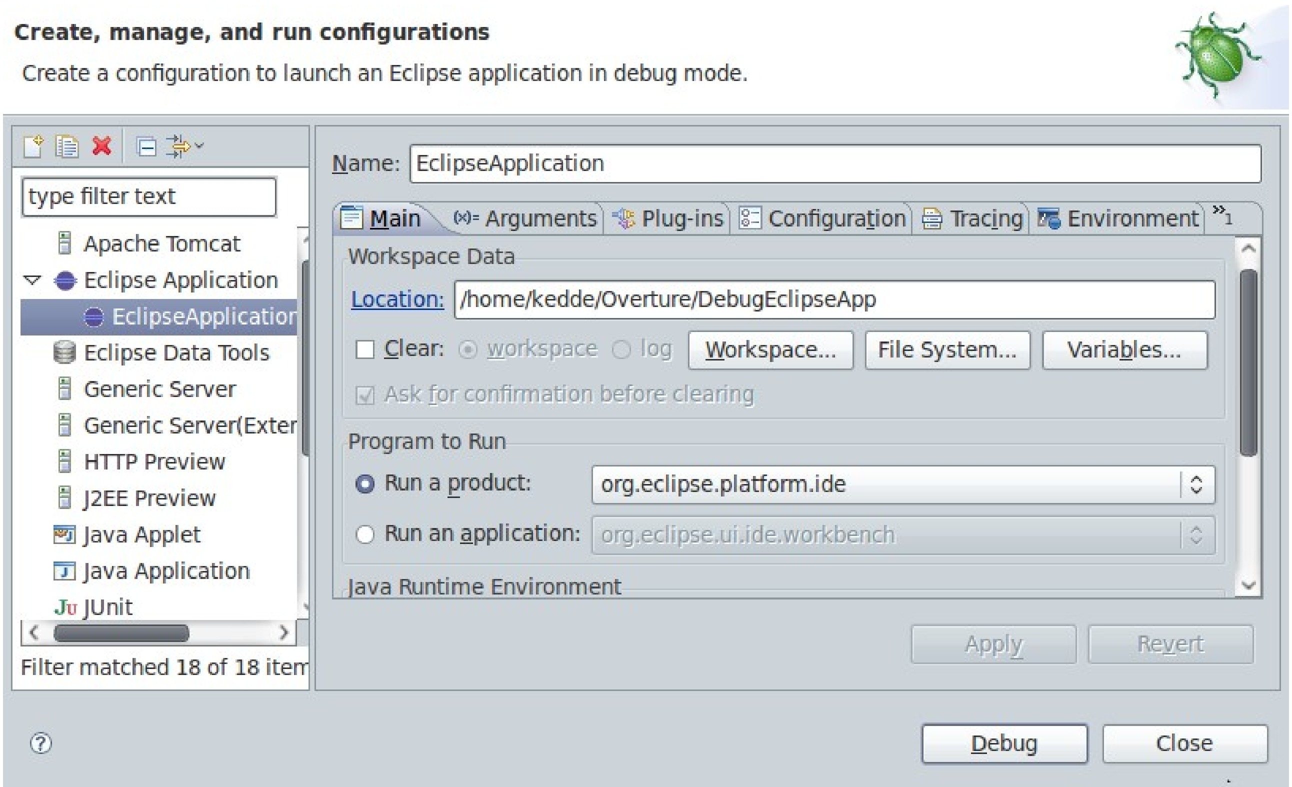
Exercise 1:

a) Start of the Overture Editor

You should at least have all the plug-in projects in you workspace, the following figure shows the projects:



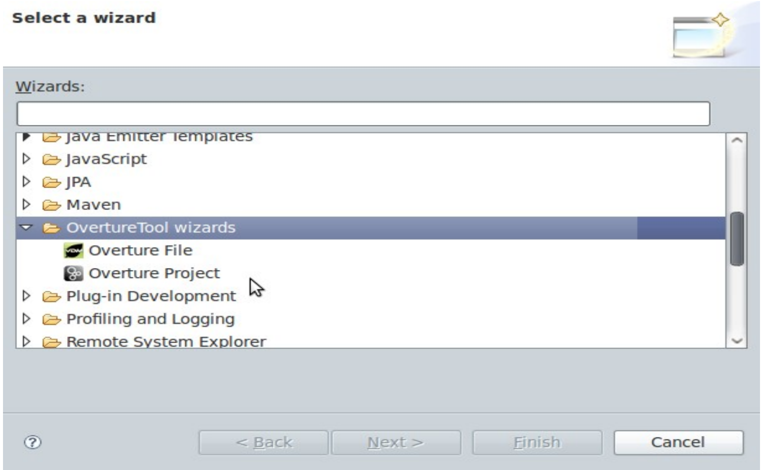
Choose the menu “Run” → “Debug Configurations...” you should now see the debug configuration dialog, where you should create a new EclipseApplication (Which is done by double clicking in the left menu). Choose a location for the workspace for the new Eclipse instance and click on the debug button.



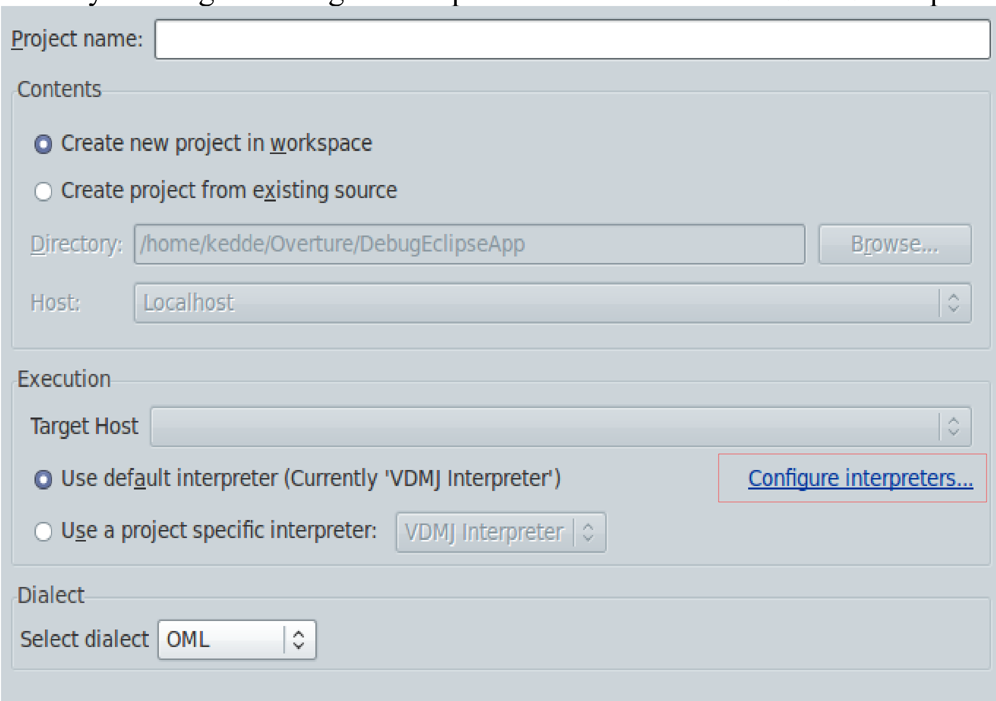
b) Debugging a VDM Model using the Overture Editor

b1) Setting up a project

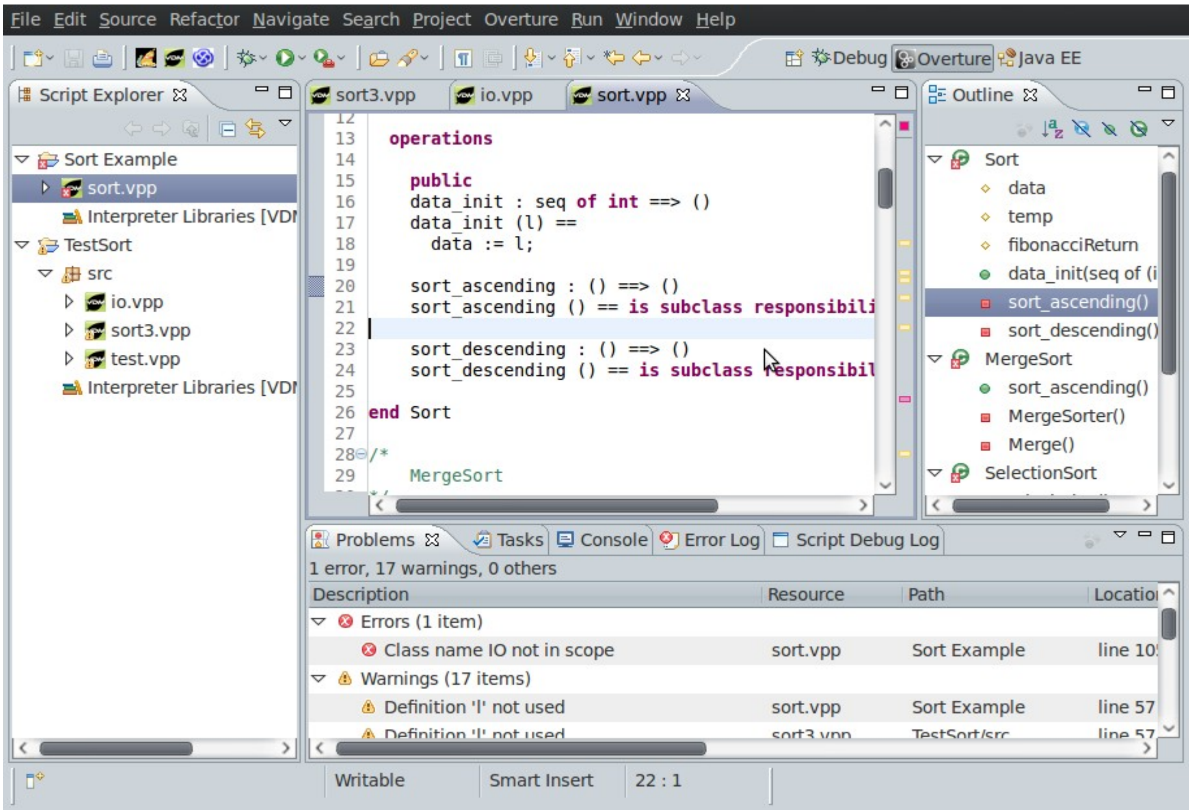
Create a new Overture Project by clicking on the and select the menu File → New → other...



The first time the user creates a Overture project the default interpreter needs to be set up. This is done by clicking on configure interpreters... and choose the VDMJ interpreter.



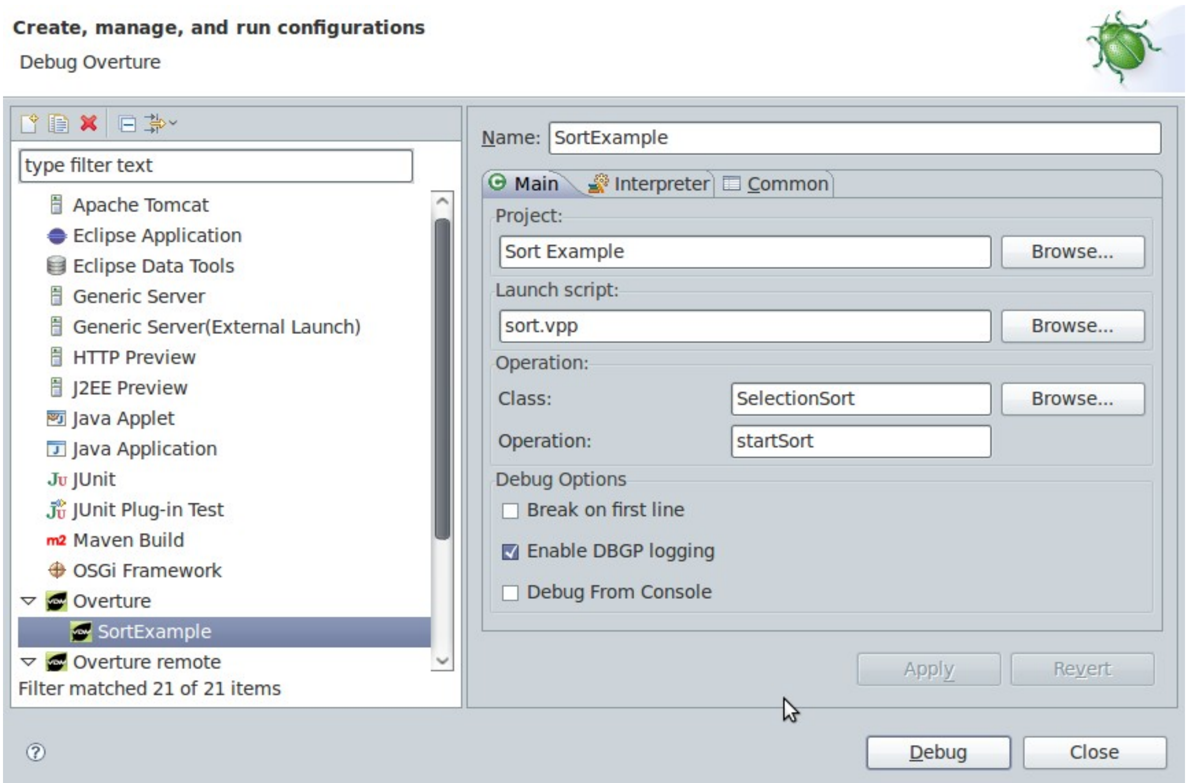
Name the new Project "Sort Example" and select finish. If everything is done properly the new project is created and is visible to the left. Add a new Overture File sort.vpp (File → new → Overture File) and copy the content from the file in the appendix to the newly created file. The editor should now look like:

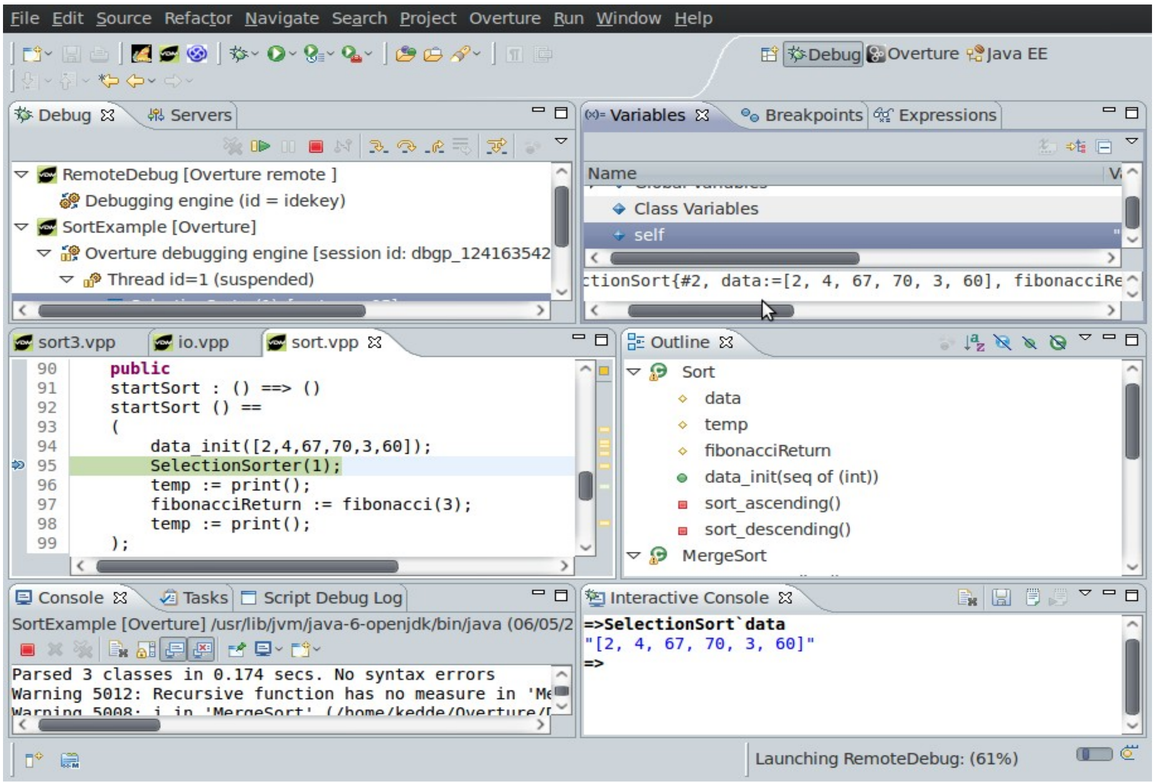


Small exercise: Double click on the error in the problem view and fix it.

b2) Debugging a VDM Model

Debugging a VDM model is similar to debugging a Java application in Eclipse. There is one addition that you need to select a "void" operation/function the interpreter going to evaluate.





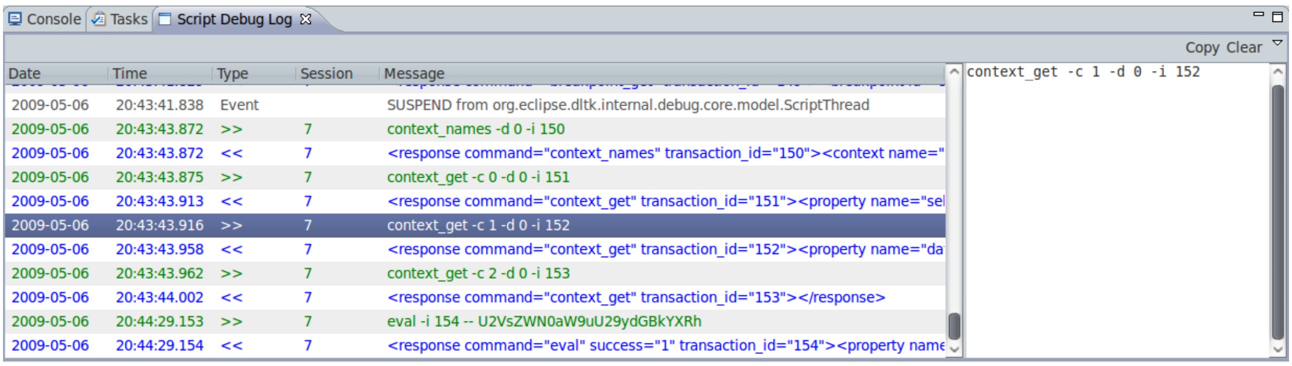
At a breakpoint you can use the either the Variables view values of variables in a given context (upper right) and you can use the interactive console to evaluate expression(“Window“ → “show view” → “Interactive console” lower right on the above picture). To step through the model it is possible to use the menu “run” → “step” . Over”.

Exercise:

Set a breakpoint at line 95 (SelectionSorter(1)) by right-click in the left margin at line 95 and select Toggle Breakpoint, a blue dot will appear in the left margin.
Debug the model with the options shown above. Now observe the variable “data” before and after the operation call.

b3) DBGP

If you are interested in the communication between the IDE and the debugger server it is possible to view this, by first selecting “enable dbgp logging” in the debug configuration dialog and afterwards show the “script debug log” view (window → show view → other... → Script debug log)



Exercise 2:

a) Syntax highlight the keyword

Adding a new keyword is quite simple, open the class OvertureCodeScanner located in the project org.overturetool.eclipse.plugins.editor.ui. And add the wanted keyword.

b) Wizard

b1) wizard

To create a wizard page similar to the above we first need to create the content of the dialog, which is done by extending WizardPage. Use the following code snippet and insert the label and the combobox.

The new class should be located in org.overturetool.eclipse.plugins.editor.internal.ui.wizards in the project called org.overturetool.eclipse.plugins.editor.ui

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;

import org.eclipse.jface.wizard.WizardPage;
import org.eclipse.swt.widgets.Combo;

public class OvertureGenerateCodeCppWizardPage extends WizardPage{
	Combo combo;
	IProject[] iprojects;

	protected OvertureGenerateCodeCppWizardPage(String pageName) {
		super(pageName);
		setTitle("Generate C++ code");
		setDescription("Please choose the wnated options");
	}

	public void createControl(Composite parent) {
		IWorkspaceRoot iworkspaceRoot =
			ResourcesPlugin.getWorkspace().getRoot();
		iprojects = iworkspaceRoot.getProjects();
	}
```

```
}
```

The wizard can now be created by extending Wizard. Again you can use a code snippet, but needs to insert the rest:

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;

import java.util.ArrayList;

import jp.co.csk.vdm.toolbox.api.corba.ToolboxAPI.ToolType;

import org.eclipse.core.resources.IContainer;
import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IFolder;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IResource;
import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.wizard.Wizard;
import org.overturetool.vdmtools.VDMToolsProject;

public class OvertureGenerateCppWizard extends Wizard {
      OvertureGenerateCodeCppWizardPage cppWizard;
      private String[] exts = new String[] { "vpp", "tex", "vdm" };

       public void addPages() {
             cppWizard = new OvertureGenerateCodeCppWizardPage("C++ page");
             addPage(cppWizard);
       }




private ArrayList<IFile> getAllMemberFiles(IContainer dir, String[] exts) {
            ArrayList<IFile> list = new ArrayList<IFile>();
            IResource[] arr = null;
            try {
                  arr = dir.members();
            } catch (CoreException e) {
            }

            for (int i = 0; arr != null && i < arr.length; i++) {
                  if (arr[i].getType() == IResource.FOLDER) {
                        list.addAll(getAllMemberFiles((IFolder) arr[i], exts));
                  }
                  else {
                        for (int j = 0; j < exts.length; j++) {
                              if
(exts[j].equalsIgnoreCase(arr[i].getFileExtension())) {
                                    list.add((IFile) arr[i]);
                                    break;
                              }
                        }
                  }
            }
            return list;
      }

      @Override
      public boolean performFinish() {
      try
      {
```

```
            ArrayList<IFile> fileNameList = getAllMemberFiles(project, exts);
            return true;
        }
        catch (Exception e) {
            return true;
        }
}

}
```

There is missing some code in the method performFinish() to determine which project member files should be code generated.

b2) action

To test the wizard, there have already been created an extension point to the menu Overture menu in the plugin.xml in the project org.overturetool.eclipse.plugins.editor.ui.

```
<extension point="org.eclipse.ui.actionSets">
     <actionSet
           id="org.overturetool.generateCpp"
           label="Generate Cpp"
           visible="true">
        <menu
              id="overtureMenu"
              label="Overture">
           <separator
                 name="slot2">
           </separator>

        </menu>
        <menu
              id="overtureCode"
              label="123">
        </menu>
           <action
      class="org.overturetool.eclipse.plugins.editor.ui.actions.GenerateCppCode"
                 id="org.overturetool.generatecpp"
                 label="Model to C++"
                 menubarPath="overtureMenu/overtureCode">
        </action>
     </actionSet>
  </extension>
```

You should create an action class that extends IWorkbenchWindowActionDelegate placed in org.overturetool.eclipse.plugins.editor.ui.actions in the run method insert the following snippet.

```
        OvertureGenerateCppWizard wizard = new OvertureGenerateCppWizard();
        WizardDialog dialog = new WizardDialog(null, wizard);
        dialog.create();
        dialog.open();
```

If you didn't name action class GenerateCppCode then you will have to rename the class attribute in the plugin.xml.

TIP: Debugging in the Overture Editor

If you didn't get the wizard right the first time, it is possible to debug the Overture Editor, just by

putting a breakpoint e.g. in the run method in the action class.
Another way to detect errors is to look in the error log in the overture editor(window → show view → other → error log)

Exercise 3: Calling code from VDMTools

To access the VDMTools Api ther in our wizard, we make use of a binary plug-in project, which is already set up for you in the manifest file.

To finish the wizard these two snippet might be helpful.

```
VDMToolsProject vdmProject = VDMToolsProject.getInstance();
                    vdmProject.init("/home/kedde/Programs/vice/bin/vicegde",
ToolType.PP_TOOLBOX);
```

```
vdmProject.addFilesToProject(filenamesString);
vdmProject.codeGenerateCPP(false);
```

Appendix

SortExample.vpp

```
--
-- class Sort
--
-- Consist
--
class Sort

  instance variables
    protected data : seq of int := [0];
    protected temp : bool := false;
    protected fibonacciReturn : nat := 0;

  operations

    public
    data_init : seq of int ==> ()
    data_init (l) ==
      data := l;

    sort_ascending : () ==> ()
    sort_ascending () == is subclass responsibility;

    sort_descending : () ==> ()
    sort_descending () == is subclass responsibility

end Sort

/*
      MergeSort
*/
class MergeSort is subclass of Sort


operations
  public
  sort_ascending : () ==> ()
  sort_ascending()   ==
    data := MergeSorter(data)

functions

  MergeSorter: seq of real -> seq of real
  MergeSorter(l) ==
    cases l:
      []       -> l,
--       [e]      -> l,
      others   -> let l1^l2 in set {l} be st abs (len l1 - len l2) < 2
                  in
                    let l_l = MergeSorter(l1),
                        l_r = MergeSorter(l2) in
                     Merge(l_l, l_r)
    end;

  Merge: seq of int * seq of int -> seq of int
  Merge(l1,l2) ==
    cases mk_(l1,l2):
      mk_([],l),mk_(l,[]) -> l,
```

```
      others                -> if hd l1 <= hd l2 then
                                  [hd l1] ^ Merge(tl l1, l2)
                               else
                                  [hd l2] ^ Merge(l1, tl l2)
    end
  pre forall i in set inds l1 & l1(i) >= 0 and
      forall i in set inds l2 & l2(i) >= 0


end MergeSort

/*
      SelectionSort
*/
class SelectionSort is subclass of Sort

  functions

    min_index : seq1 of nat -> nat
    min_index(l) ==
      if len l = 1
      then 1
      else let mi = min_index(tl l)
           in if l(mi+1) < hd l
              then mi+1
              else 1


  operations
    /*
    * StartSort testing :
    */
      public
    startSort : () ==> ()
    startSort () ==
    (
      data_init([2,4,67,70,3,60]);
            SelectionSorter(1);
            temp := print();
      fibonacciReturn := fibonacci(3);
      temp := print();
    );

      private print : () ==> bool
      print () ==
      (
            return false;
            --return new IO().echo("Hello world!\n");
      );

      private
    sort_ascending : () ==> ()
    sort_ascending () == SelectionSorter(1);
      /*
            SelectionSorter
      */
      private
    SelectionSorter : nat ==> ()
    SelectionSorter (i) ==
      if i < len data
      then (
                  dcl temp1: nat;
            dcl mi : nat := min_index(data(len data,...,len data)) + i - 1;
```

```
            temp1 := data(mi);
            data(mi) := data(i);
            data(i) := temp1;
            SelectionSorter(i+1)
           )

functions
      public fibonacci: nat -> nat
            fibonacci(x) ==
            (
                  if x = 1
                        then 1
                        else x * fibonacci(x-1)
            );

end SelectionSort
```

Solution to selectionSorter

```
private
  SelectionSorter : nat ==> ()
  SelectionSorter (i) ==
   if i < len data
   then (dcl temp1: nat;
      dcl mi : nat := min_index(data(i,...,len data)) + i - 1;
      temp1 := data(mi);
      data(mi) := data(i);
      data(i) := temp1;
      SelectionSorter(i+1)
     )

correct is
min_index(data(i,...,len data)) + i - 1;
```

OvertureGenerateCppWizard

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;

import java.util.ArrayList;

import jp.co.csk.vdm.toolbox.api.corba.ToolboxAPI.ToolType;

import org.eclipse.core.resources.IContainer;
import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IFolder;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IResource;
import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.wizard.Wizard;
import org.overturetool.vdmtools.VDMToolsProject;

public class OvertureGenerateCppWizard extends Wizard {
      OvertureGenerateCodeCppWizardPage cppWizard;
      private String[] exts = new String[] { "vpp", "tex", "vdm" };

       public void addPages() {
```

```
			cppWizard = new OvertureGenerateCodeCppWizardPage("C++ page");
			addPage(cppWizard);
		}

		/**
		 * This method returns a list of files under the given directory or its
		 * subdirectories. The directories themselves are not returned.
		 *
		 * @param dir
		 *            a directory
		 * @return list of IResource objects representing the files under the
given
		 *         directory and its subdirectories
		 */
	private ArrayList<IFile> getAllMemberFiles(IContainer dir, String[] exts)
{
		ArrayList<IFile> list = new ArrayList<IFile>();
		IResource[] arr = null;
		try {
			arr = dir.members();
		} catch (CoreException e) {
		}

		for (int i = 0; arr != null && i < arr.length; i++) {
			if (arr[i].getType() == IResource.FOLDER) {
				list.addAll(getAllMemberFiles((IFolder) arr[i], exts));
			}
			else {
				for (int j = 0; j < exts.length; j++) {
					if
(exts[j].equalsIgnoreCase(arr[i].getFileExtension())) {
						list.add((IFile) arr[i]);
						break;
					}
				}
			}
		}
		return list;
	}

	@Override
	public boolean performFinish() {
		try
		{
			VDMToolsProject vdmProject = VDMToolsProject.getInstance();
			vdmProject.init("/home/kedde/Programs/vice/bin/vicegde",
ToolType.PP_TOOLBOX);
			IProject proj = null;
			for (IProject project : cppWizard.iprojects) {
				if
(cppWizard.combo.getText().equals(project.getName())){
					proj = project;
				}
			}
			ArrayList<IFile> fileNameList = getAllMemberFiles(proj,
exts);
			ArrayList<String> filenamesString = new ArrayList<String>();
			for (IFile file : fileNameList) {
				filenamesString.add(file.getLocationURI().getPath());
			}
			vdmProject.addFilesToProject(filenamesString);
			vdmProject.codeGenerateCPP(false);
			return true;
```

```
            }
            catch (Exception e) {
                  return true;
            }

      }

}
```

GenerateCppCode

```
package org.overturetool.eclipse.plugins.editor.ui.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.wizard.WizardDialog;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import
org.overturetool.eclipse.plugins.editor.internal.ui.wizards.OvertureGenerateCpp
Wizard;

public class GenerateCppCode implements IWorkbenchWindowActionDelegate {

      public void dispose() {

      }

      public void init(IWorkbenchWindow arg0) {

      }

      public void run(IAction arg0) {
            OvertureGenerateCppWizard wizard = new OvertureGenerateCppWizard();
        WizardDialog dialog = new WizardDialog(null, wizard);
        dialog.create();
        dialog.open();

      }

      public void selectionChanged(IAction arg0, ISelection arg1) {

      }

}
```

OvertureGenerateCodeCppWizardPage

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;

import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IWorkspaceRoot;
import org.eclipse.core.resources.ResourcesPlugin;
import org.eclipse.jface.wizard.WizardPage;
import org.eclipse.swt.SWT;
```

```java
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Combo;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;

public class OvertureGenerateCodeCppWizardPage extends WizardPage{
	Combo combo;
	IProject[] iprojects;


	protected OvertureGenerateCodeCppWizardPage(String pageName) {
		super(pageName);
		setTitle("Generate C++ code");
	  setDescription("Please choose the wnated options");
	}

	public void createControl(Composite parent) {
		Composite composite = new Composite(parent, SWT.NONE);
	  GridLayout layout = new GridLayout();
	  layout.numColumns = 2;
	  composite.setLayout(layout);
	  setControl(composite);
	  new Label(composite,SWT.NONE).setText("Select project: ");
	  combo = new Combo(composite, SWT.NONE);
	  IWorkspaceRoot iworkspaceRoot =
ResourcesPlugin.getWorkspace().getRoot();
		iprojects = iworkspaceRoot.getProjects();
		for (IProject project : iprojects) {
			combo.add(project.getName());
		}
		combo.select(0);
	}


}
```