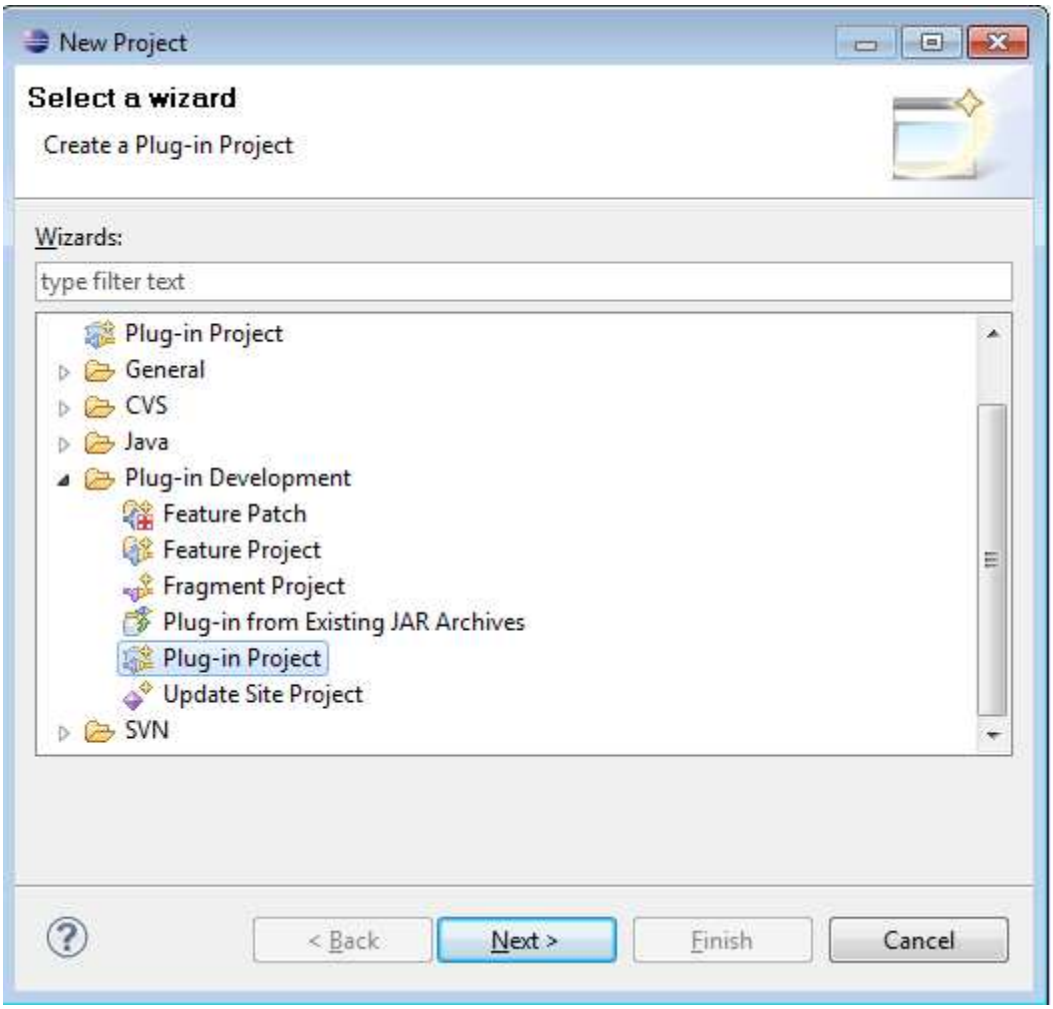
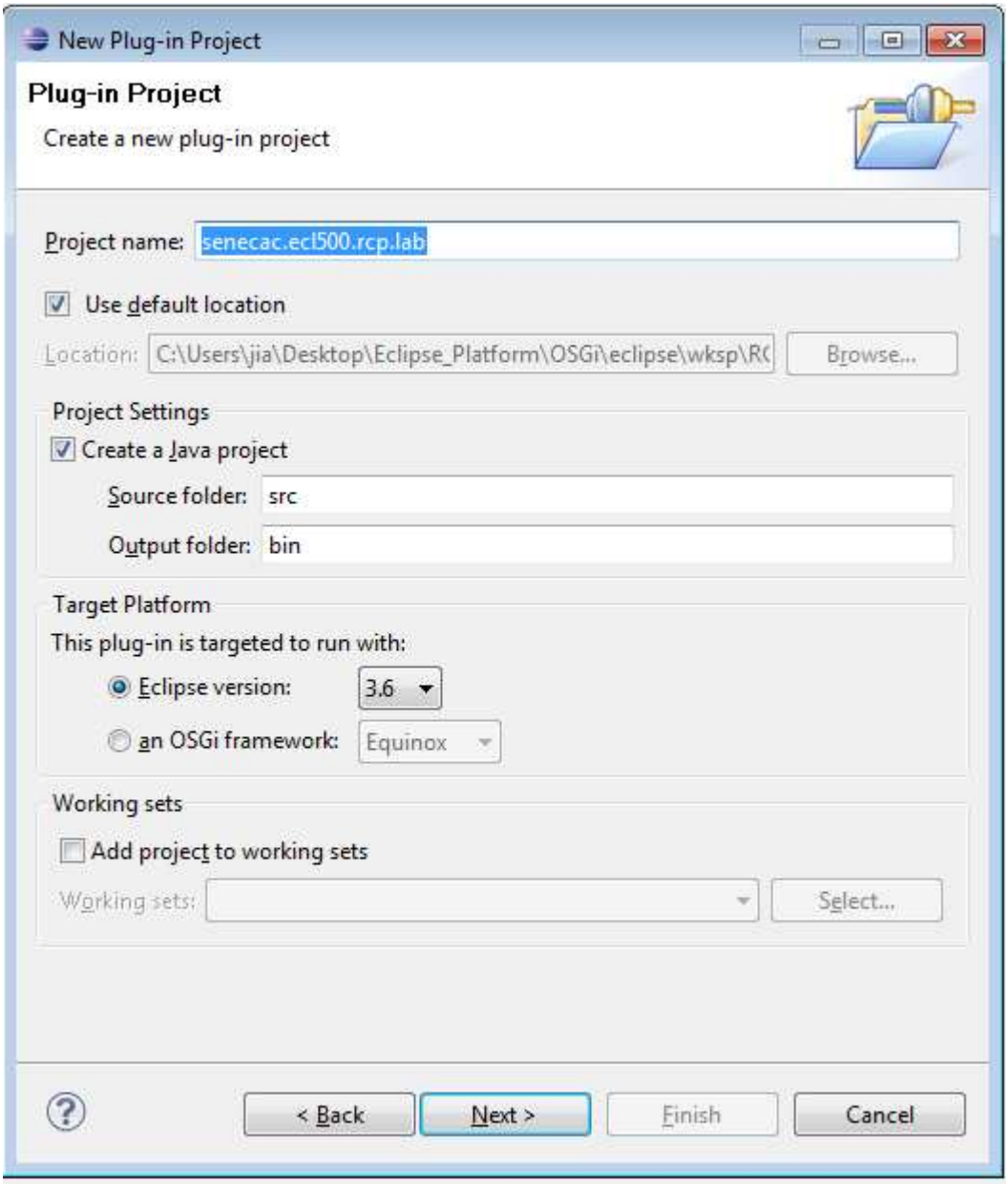
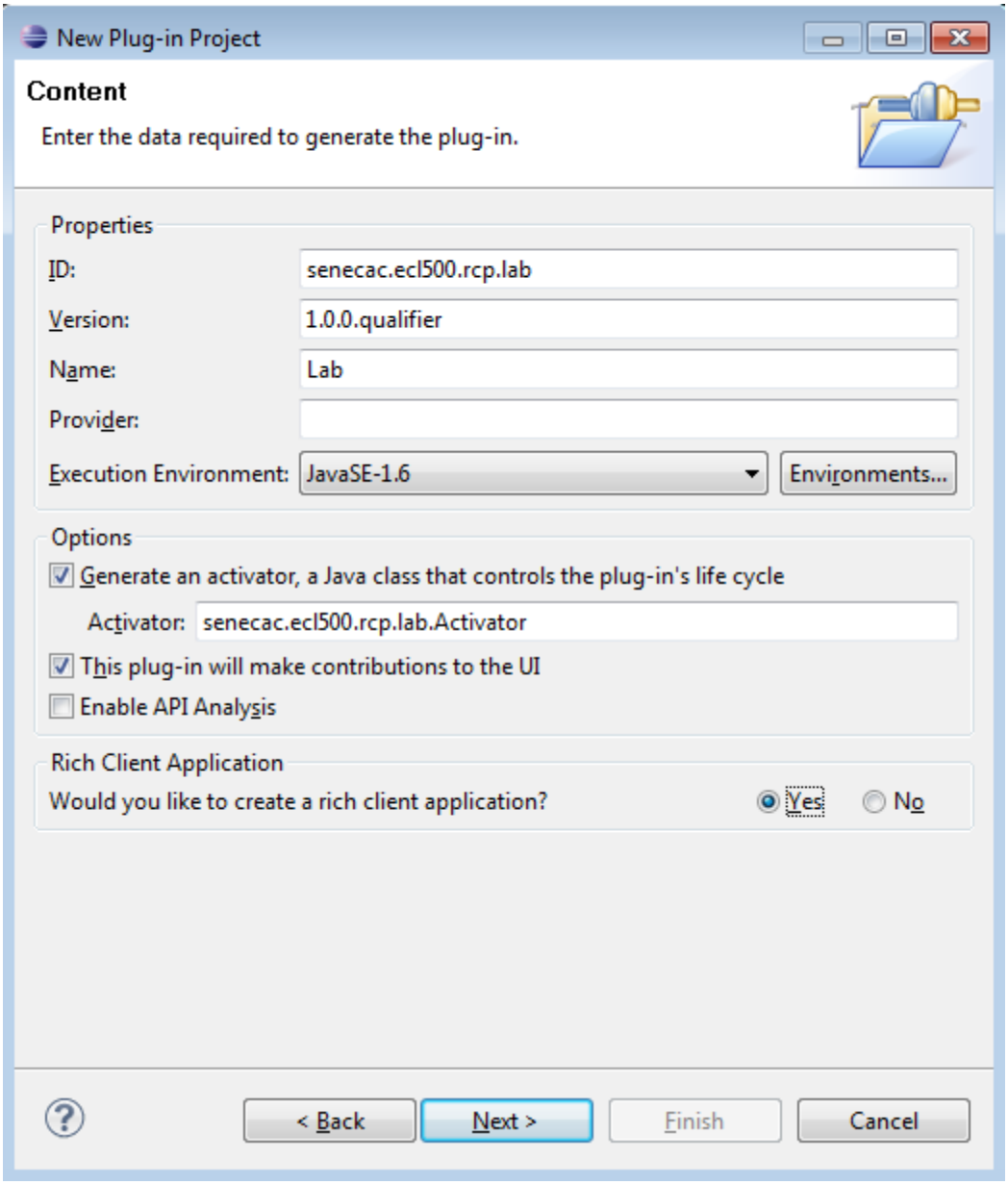
Create RCP application

Menu ’File’->’New’->’Project…’, in ‘Plug-in Development’, choose ‘Plug-In Project’



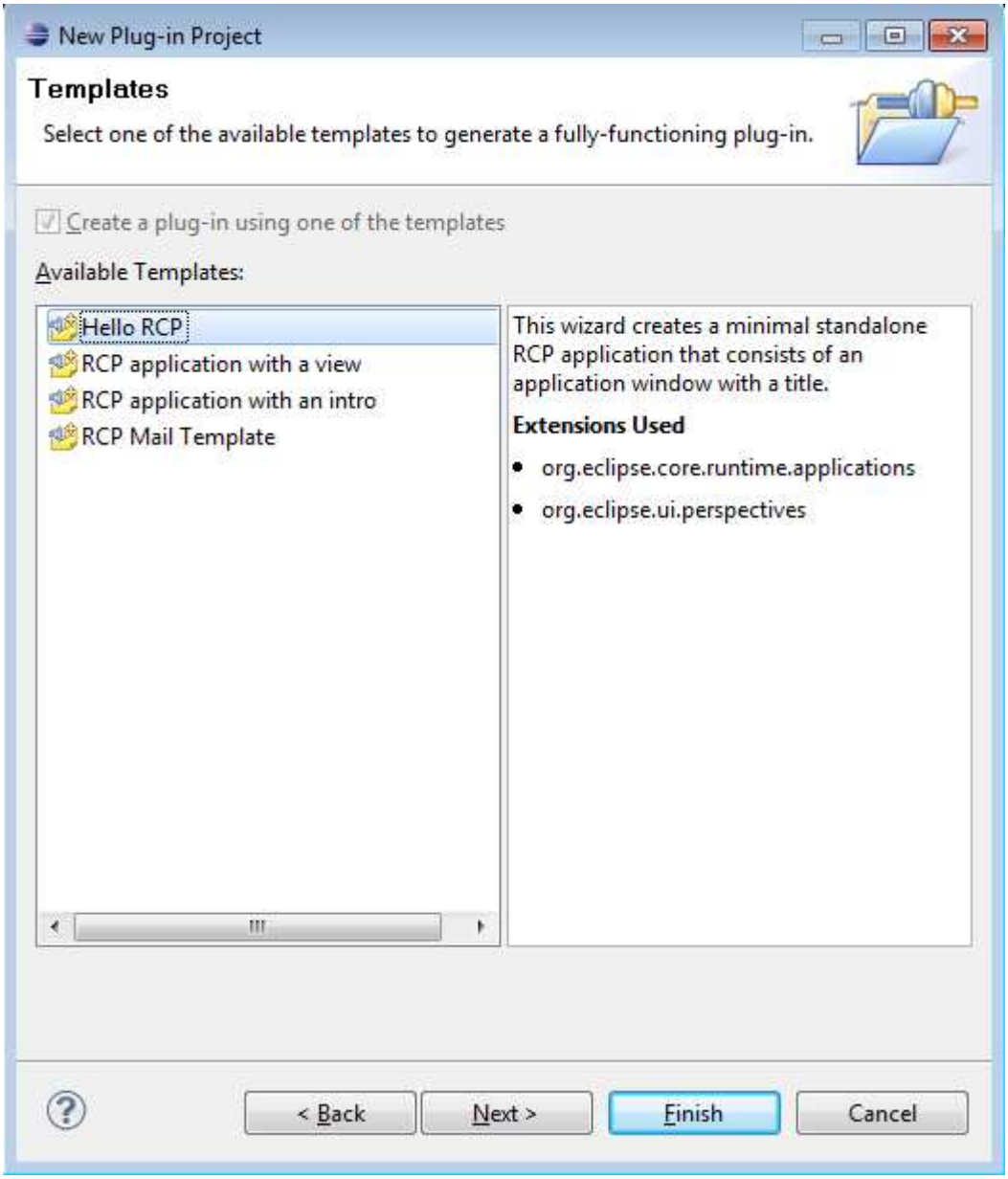
‘Next’, input a ‘Project name:’ with ‘senecac.ecl500.rcp.lab’->’Next’


New Plug-in Project
Plug-in Project
Create a new plug-in project
Project name: senecac.ecl500.rcp.lab
Use default location
Location: C:\Users\jia\Desktop\Eclipse_Platform\OSGi\eclipse\wksp\RC
Browse...
Project Settings
Create a Java project
Source folder: src
Output folder: bin
Target Platform
This plug-in is targeted to run with:
Eclipse version: 3.6
an OSGi framework: Equinox
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

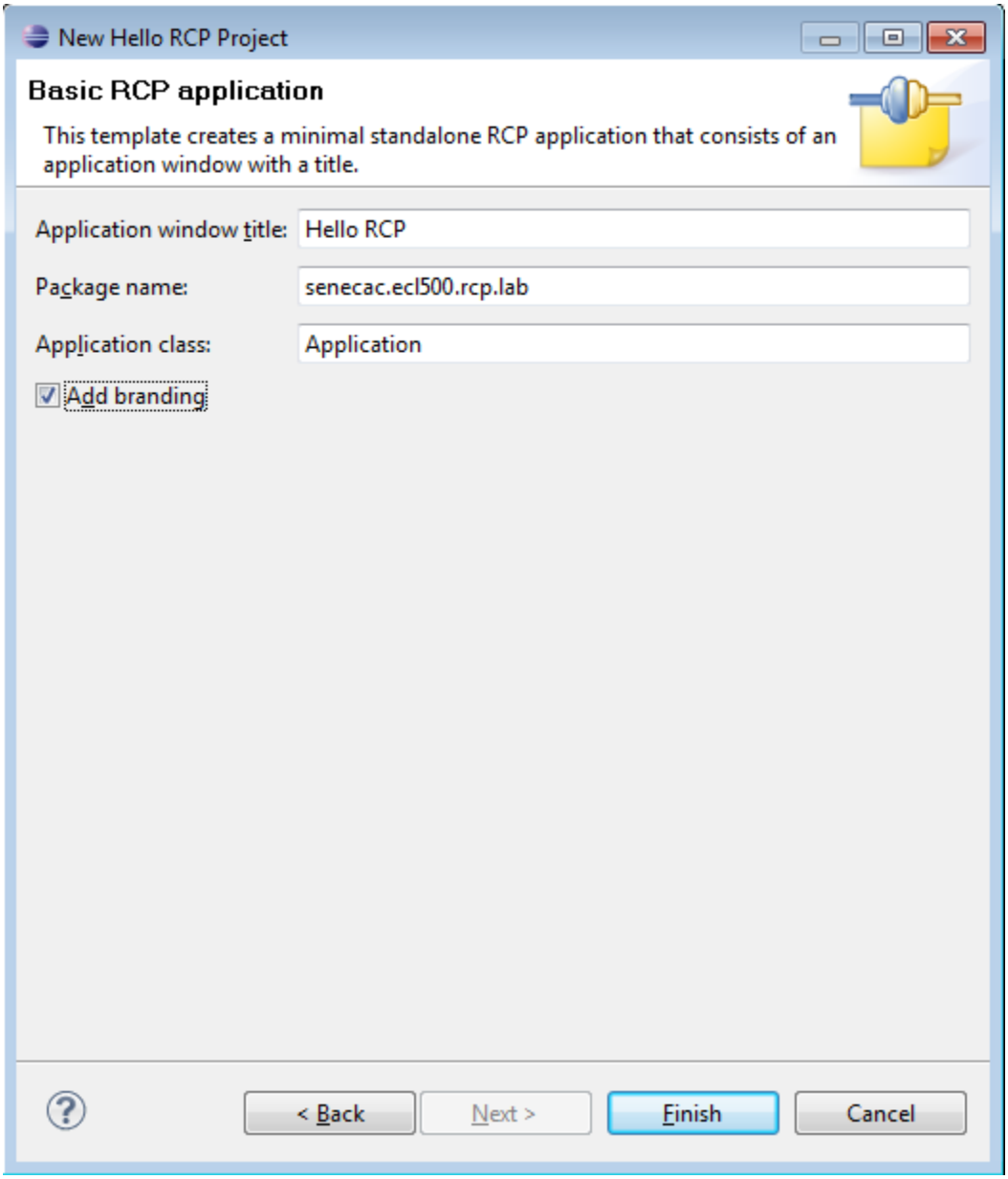


‘Next’

Choose ’Hello RCP’


New Plug-in Project
Templates
Select one of the available templates to generate a fully-functioning plug-in.
Create a plug-in using one of the templates
Available Templates:
Hello RCP
RCP application with a view
RCP application with an intro
RCP Mail Template
This wizard creates a minimal standalone RCP application that consists of an application window with a title.
Extensions Used
org.eclipse.core.runtime.applications
org.eclipse.ui.perspectives
< Back
Next >
Finish
Cancel

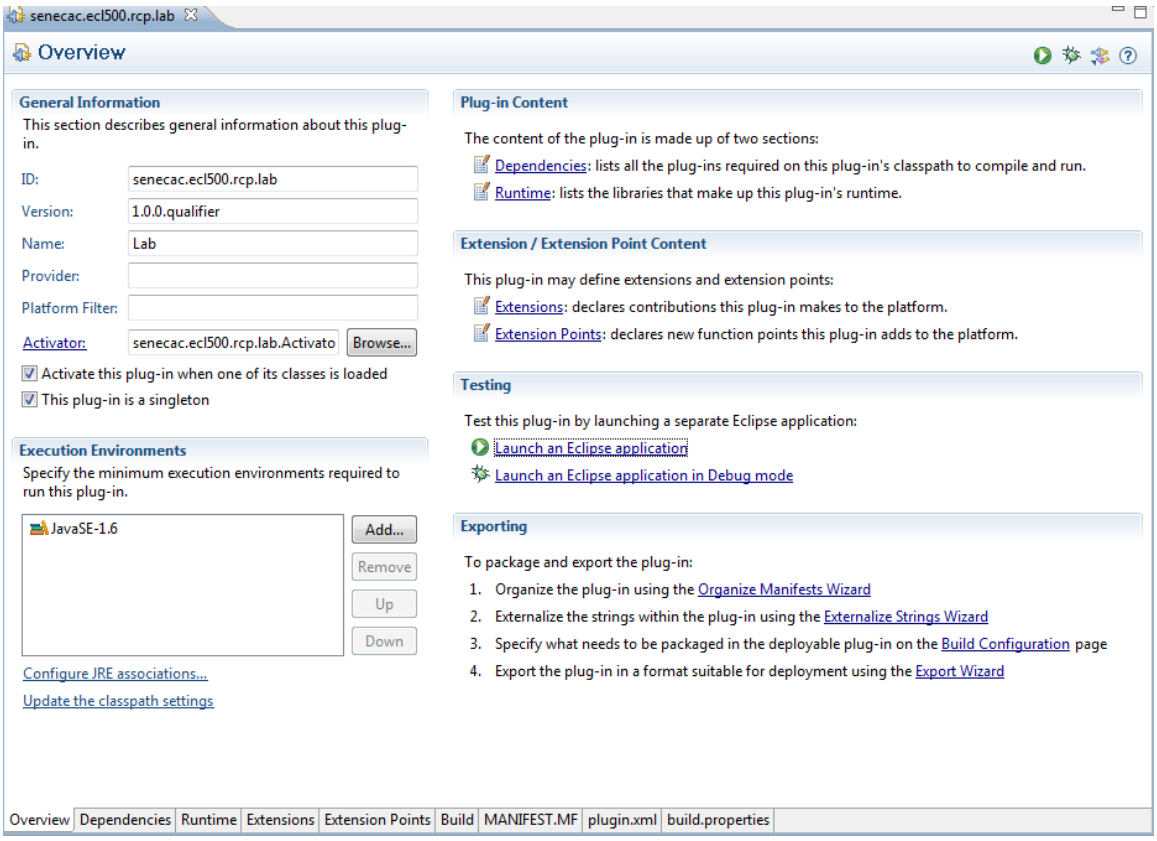
'Next'



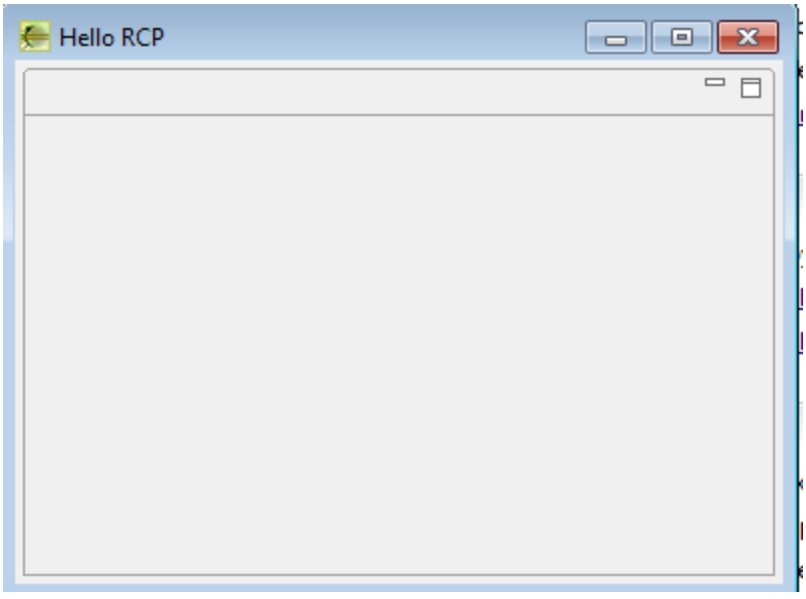
Select ‘Add branding’, ‘Finish’

Start RCP application

In ‘MAINFEST.MF’’S ‘Overview’, to start RCP application, click ‘Launch an Eclipse application’

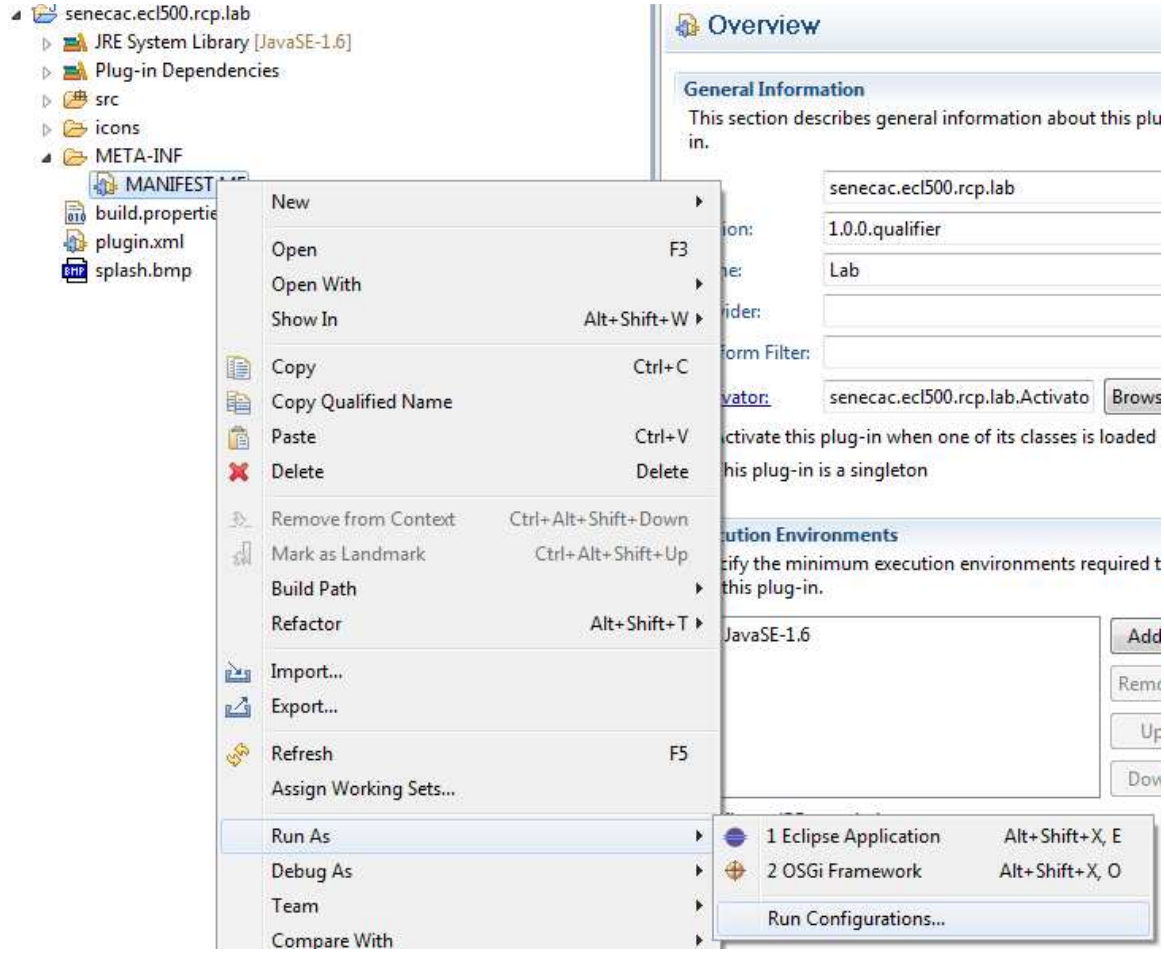


Get the window

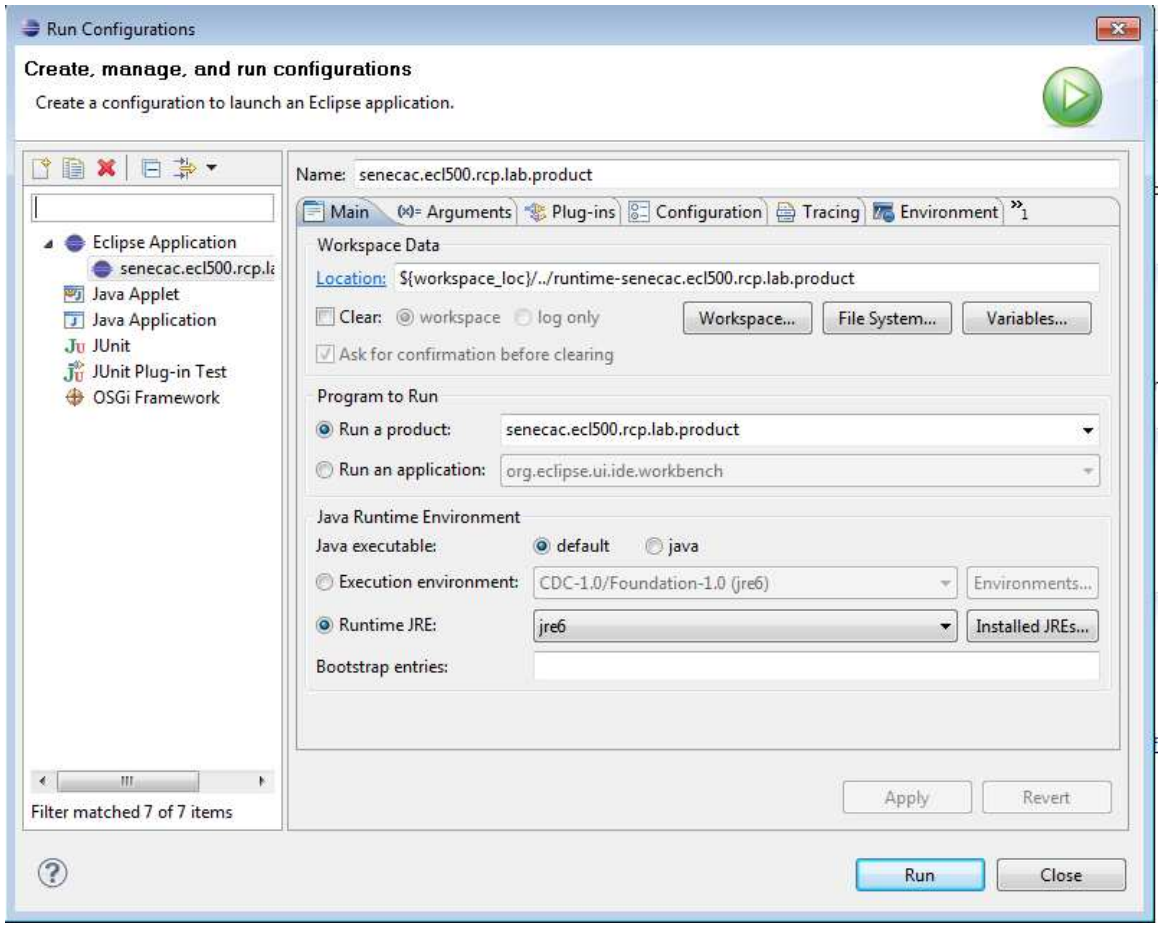


Run configuration

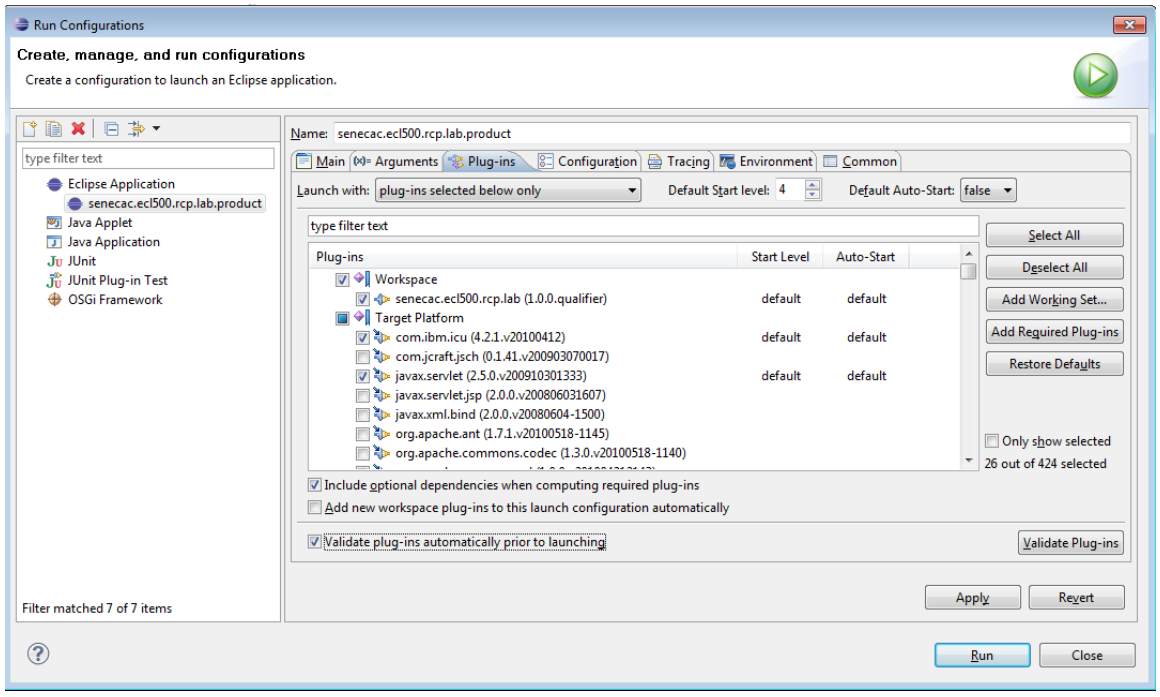
Right click at ‘MANIFEST.MF’, select ‘Run As’->’ Run configurations…’



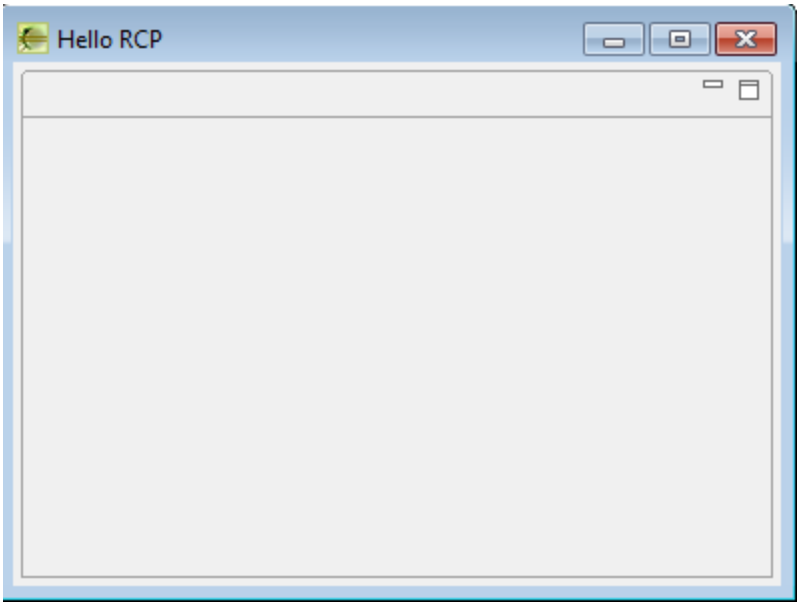
Get



Switch to ‘Plus-ins’ tag

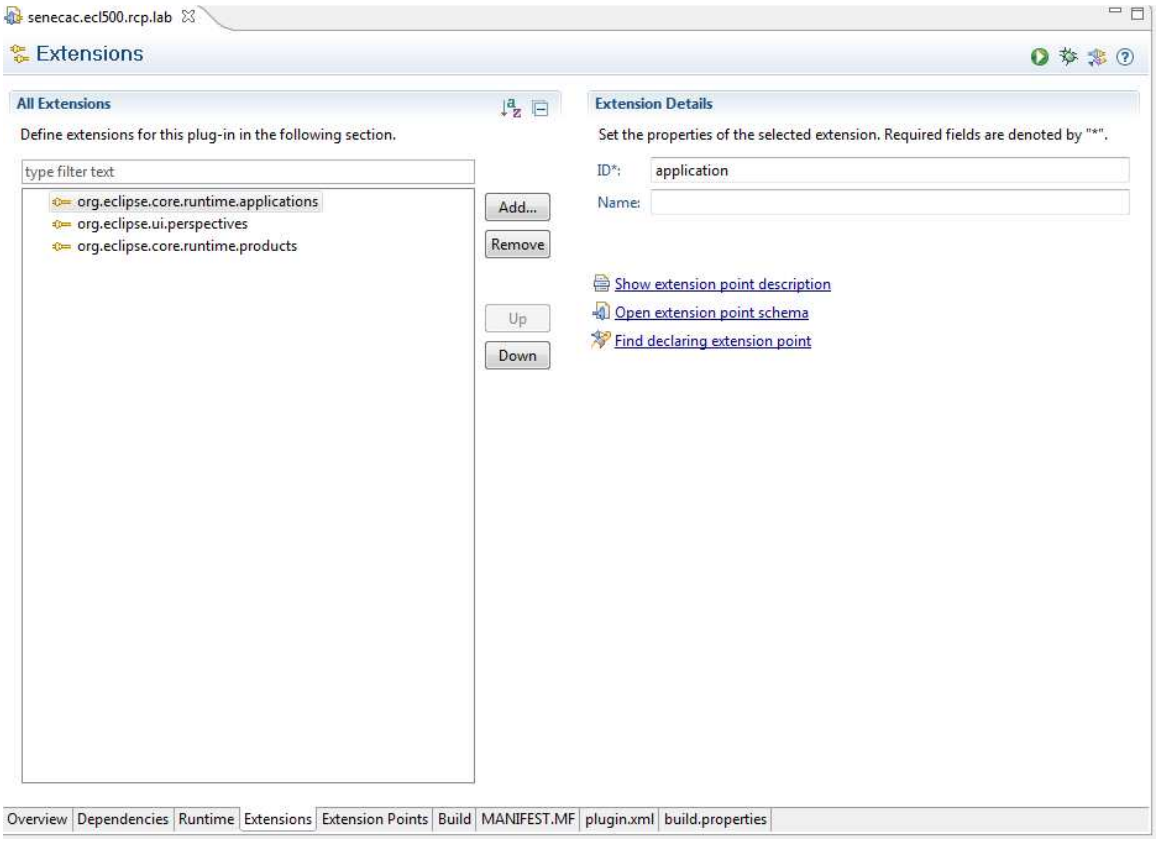


Check ‘Validate plug-ins prior to launching’, then ‘Run’, get

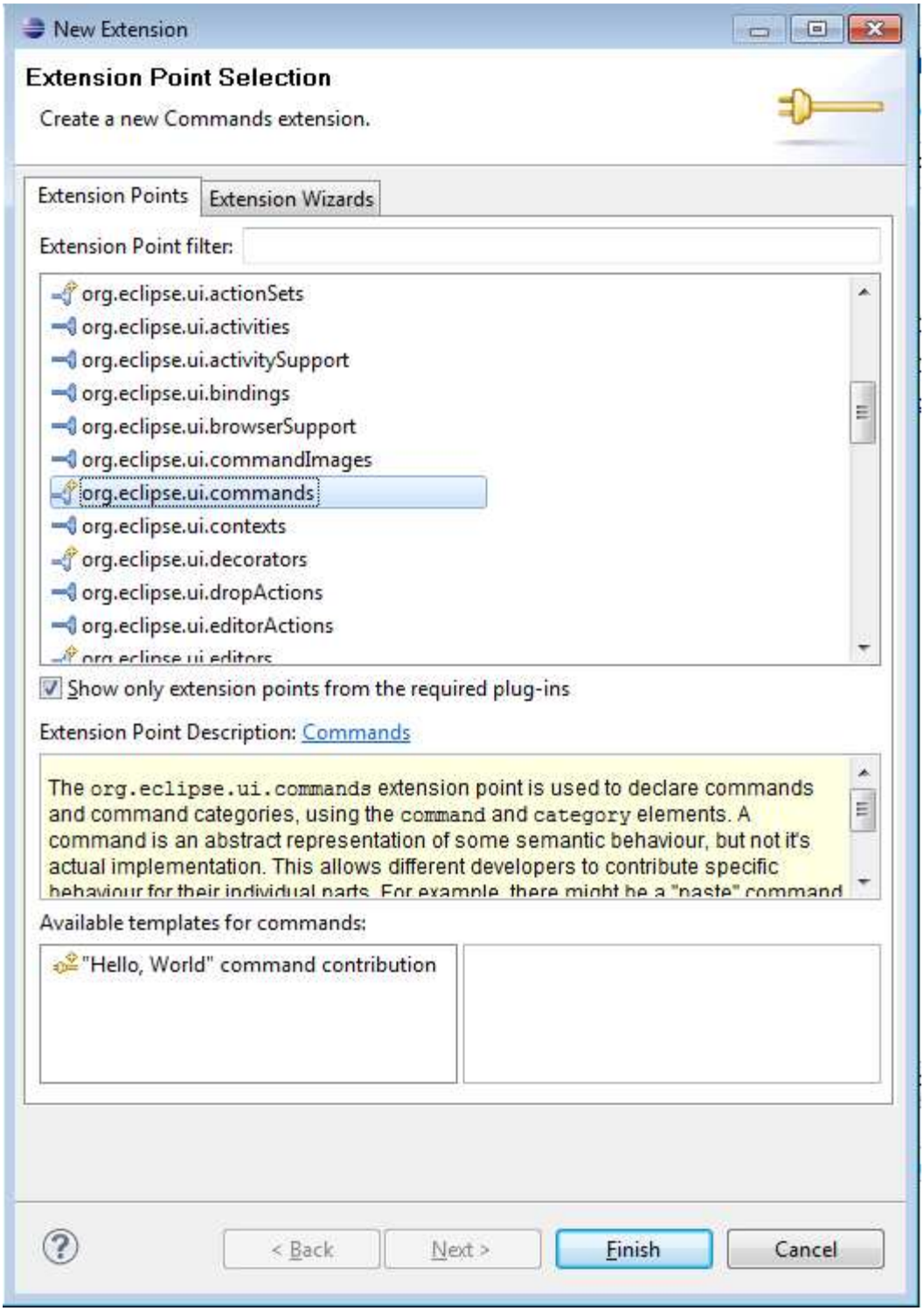


Define commands

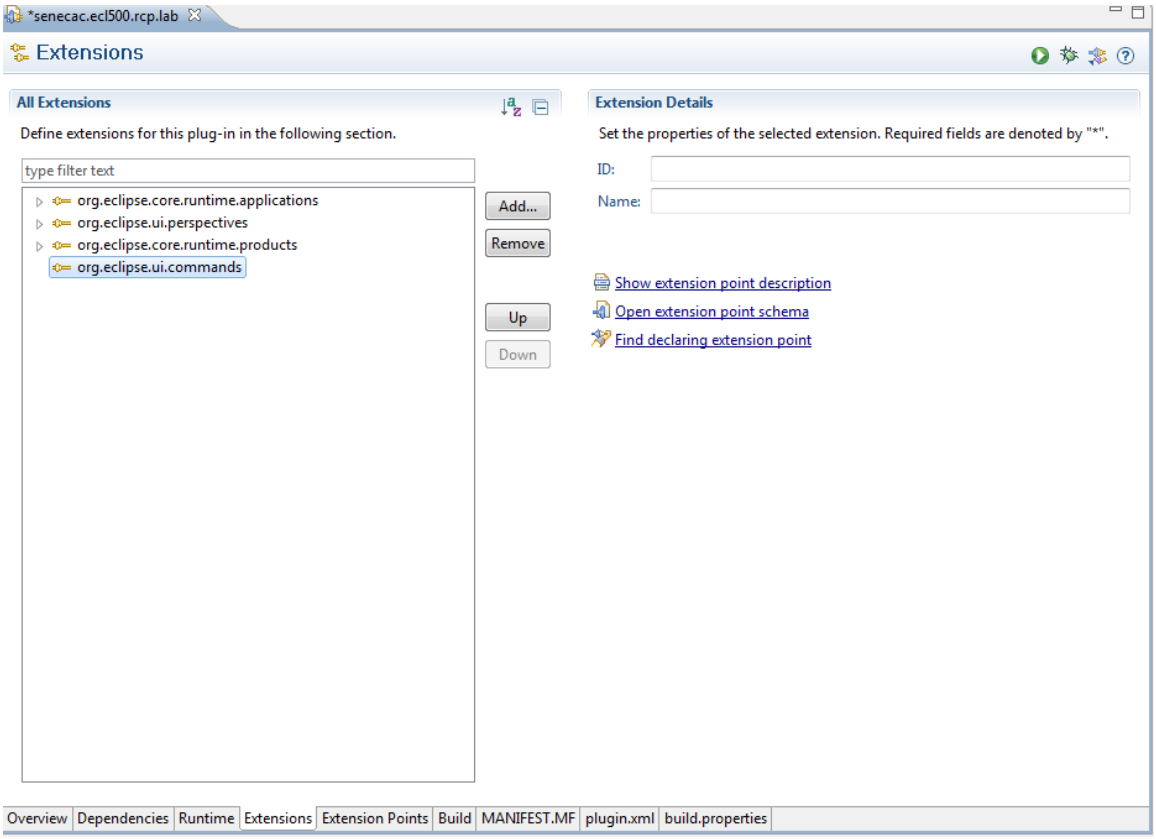
Open ‘plugin.xml’



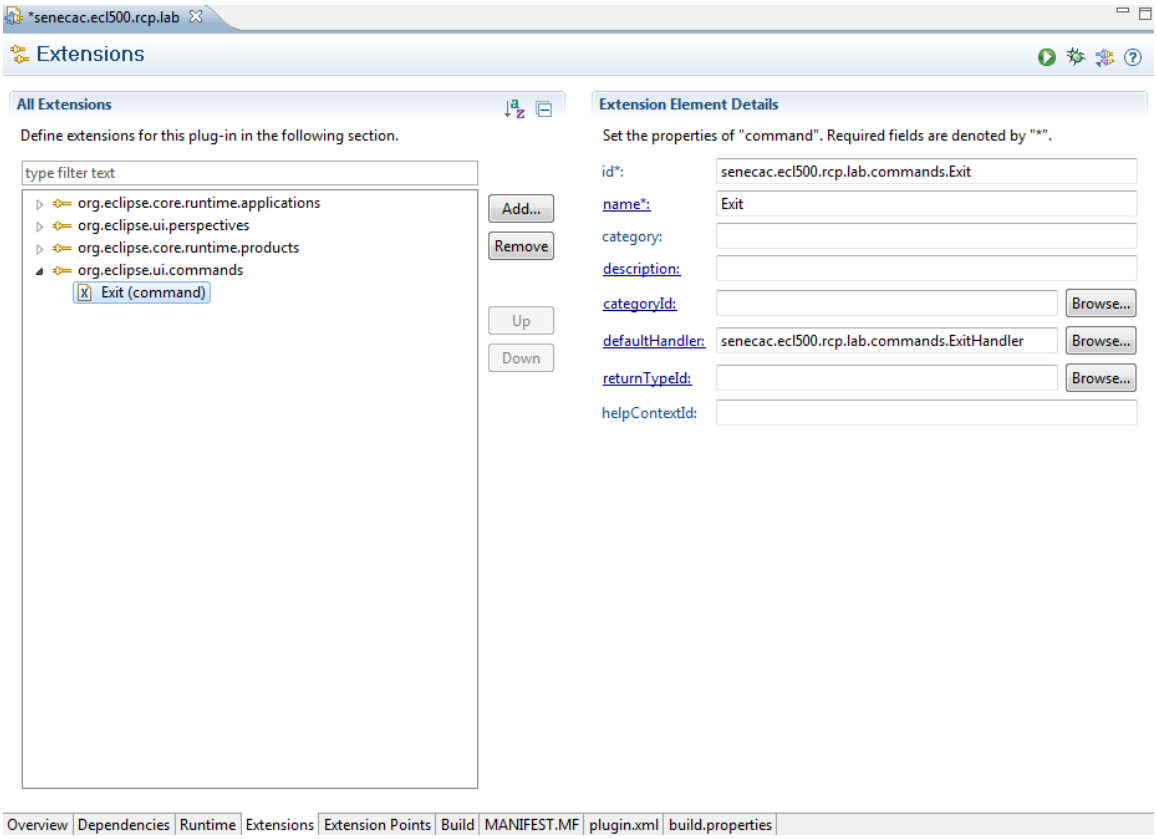
Click ‘Add’



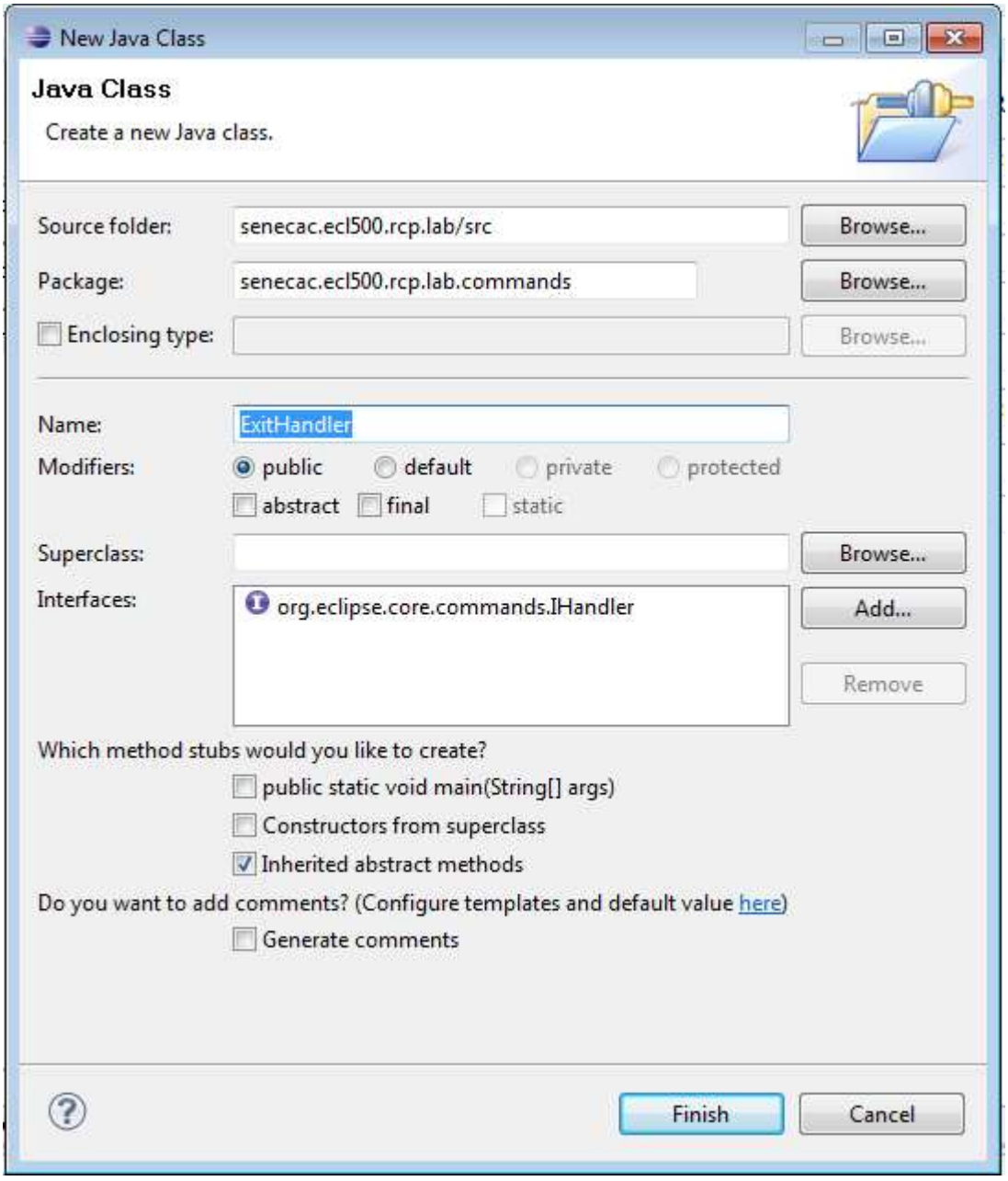
Select ‘org.eclipse.ui.commands’, Then ‘Finish’



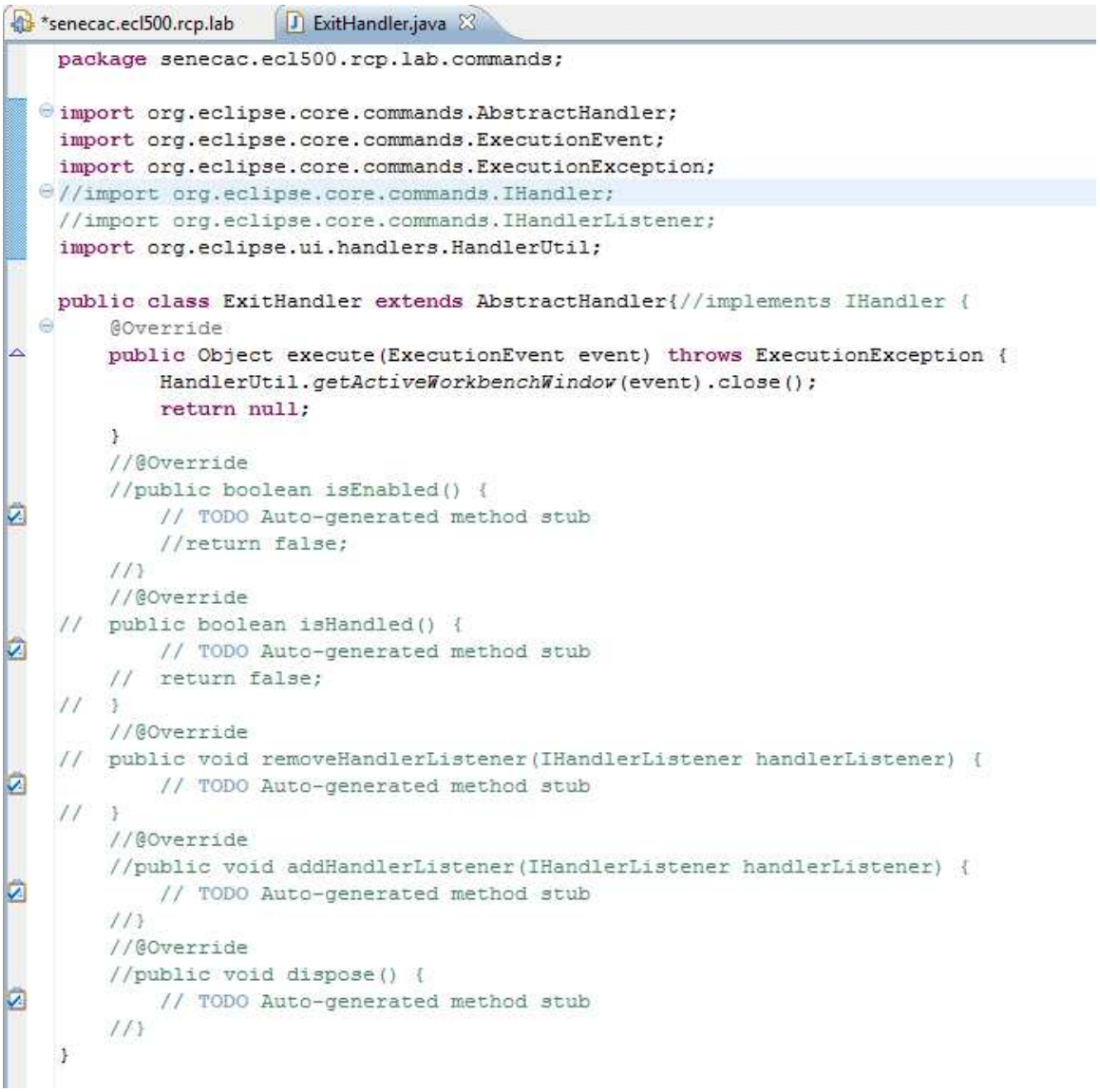
Right click ‘org.eclipse.ui.commands’->’New’->’command’



Fill ‘Name’ with ‘Exit’, ‘id*’ with ‘senecac.ecl500.rcp.lab.commands.Exit’ and ‘defaultHander’ with ‘senecac.ecl500.rcp.lab.commands.ExitHandler’, then click the link of ‘defaultHander’



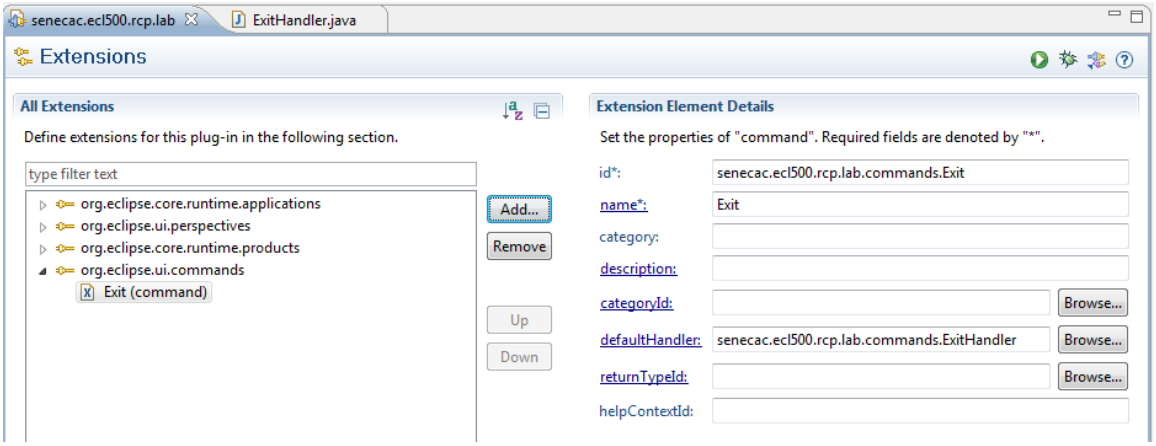
Click ‘Finish’ then edit the code as show



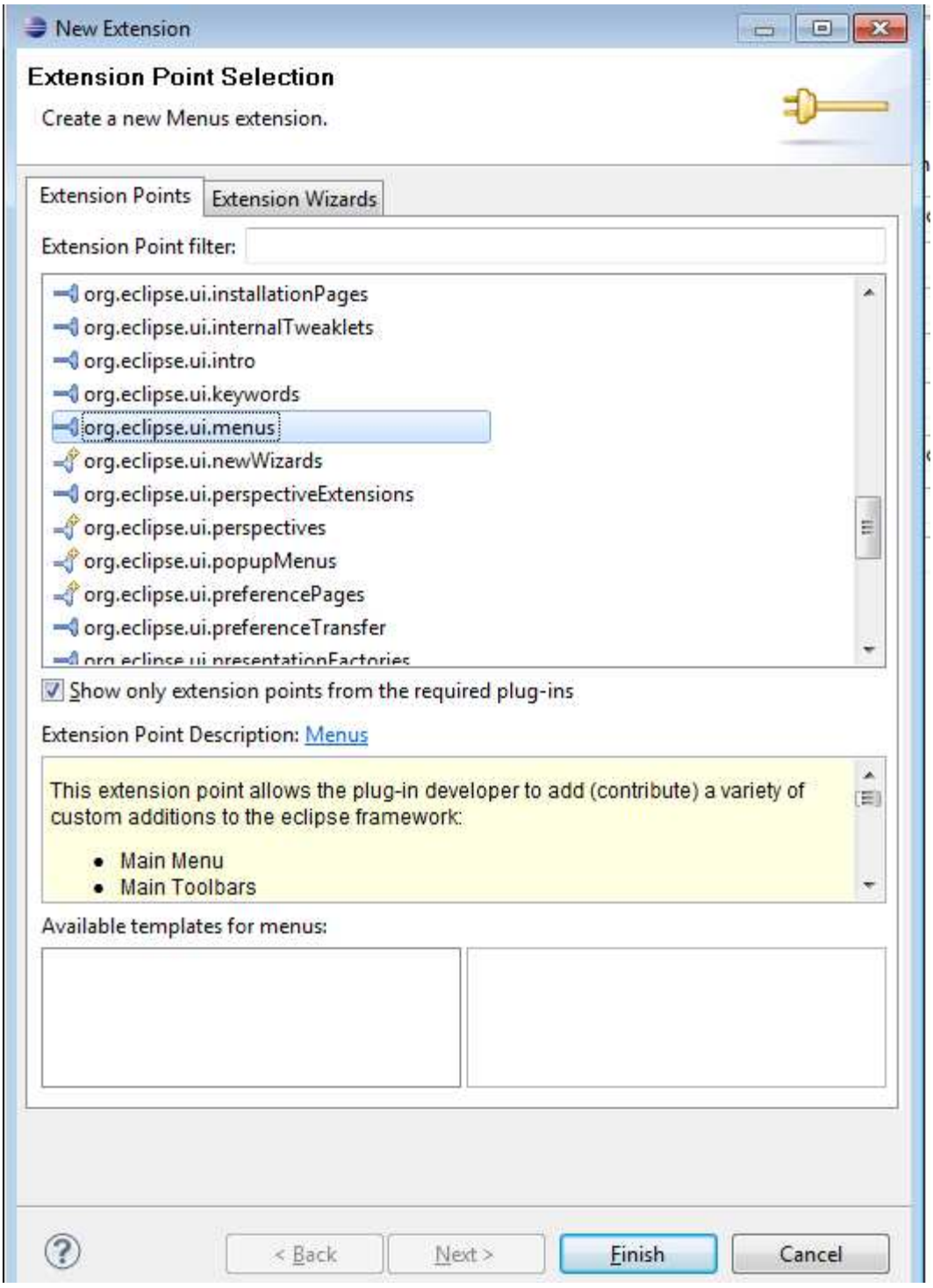
‘Save’.

Use commands

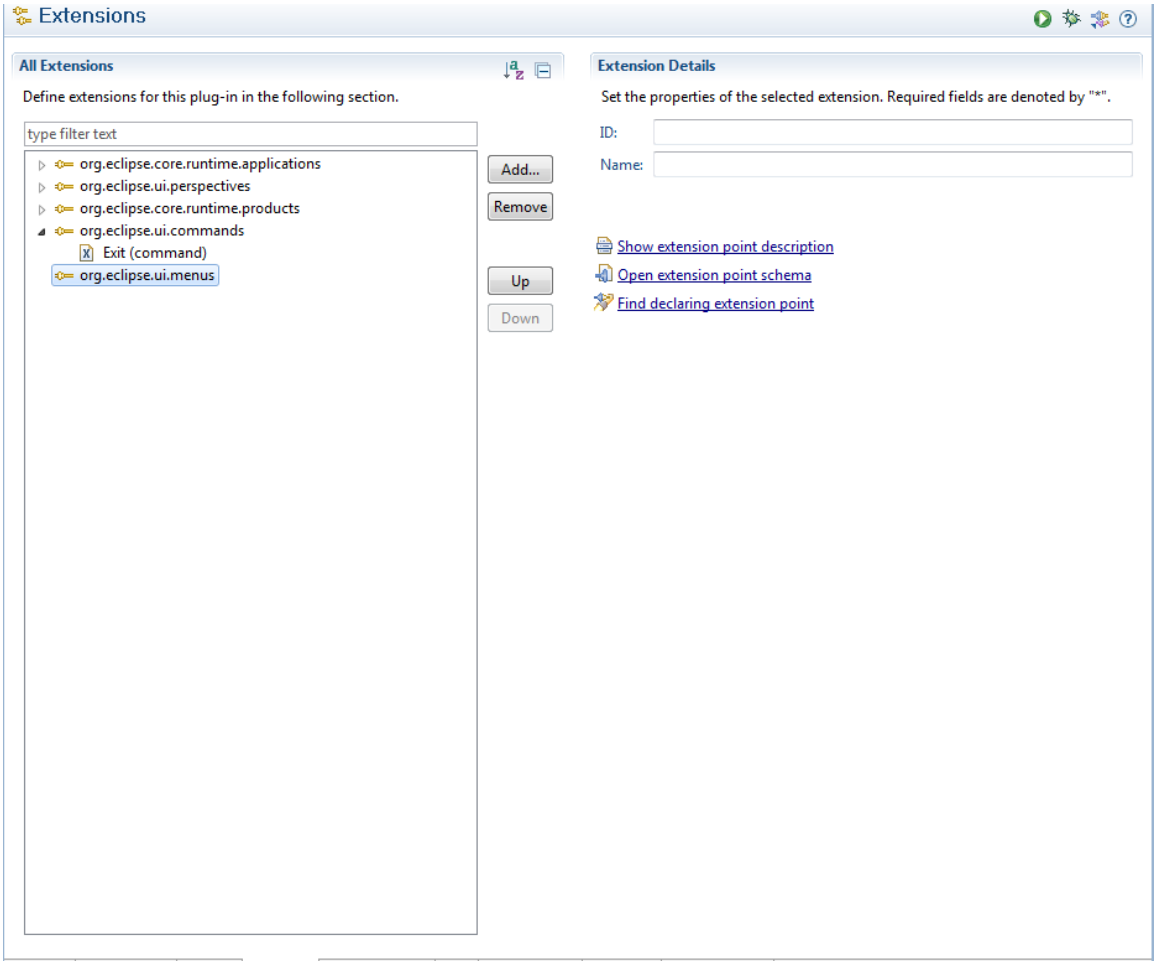
Open ‘plugin.xml’


senecac.ecl500.rcp.lab
ExitHandler.java
Extensions
All Extensions
Define extensions for this plug-in in the following section.
type filter text
org.eclipse.core.runtime.applications
org.eclipse.ui.perspectives
org.eclipse.core.runtime.products
org.eclipse.ui.commands
Exit (command)
Add...
Remove
Up
Down
Extension Element Details
Set the properties of "command". Required fields are denoted by "*".
id*:
senecac.ecl500.rcp.lab.commands.Exit
name*:
Exit
category:
description:
categoryId:
Browse...
defaultHandler:
senecac.ecl500.rcp.lab.commands.ExitHandler
Browse...
returnTypeId:
Browse...
helpContextId:

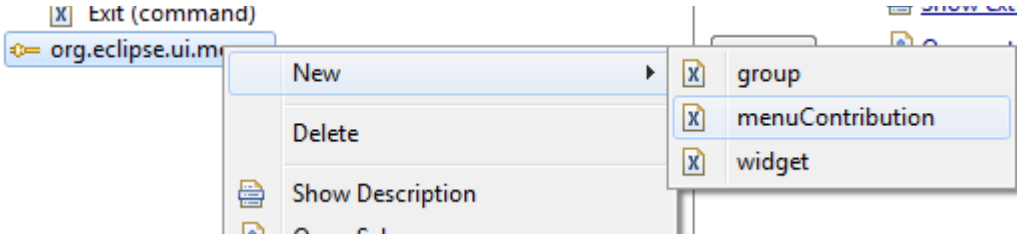
Click ‘Add…’



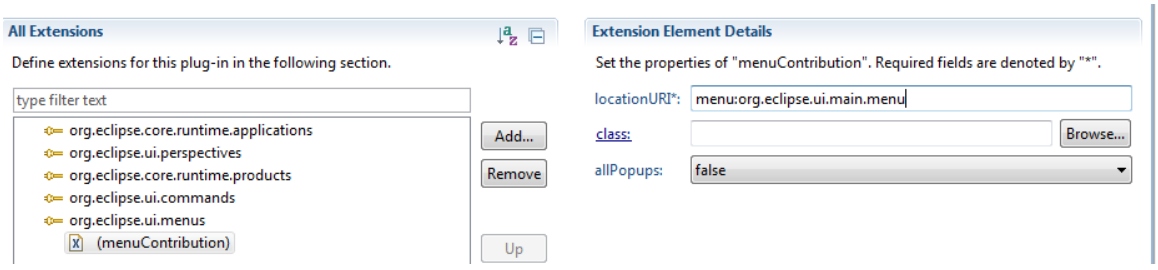
Select ‘org.eclipse.ui.menus’, ‘Finish’



Right click ‘org.eclipse.ui.menus’->’New’->‘ menuContribution’

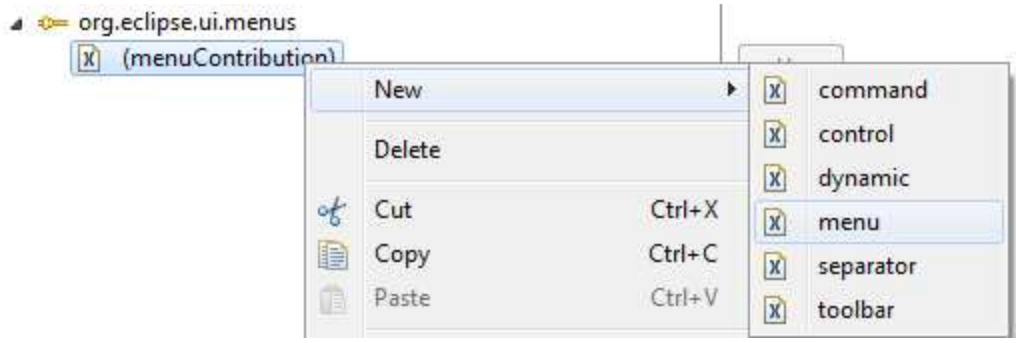


Get

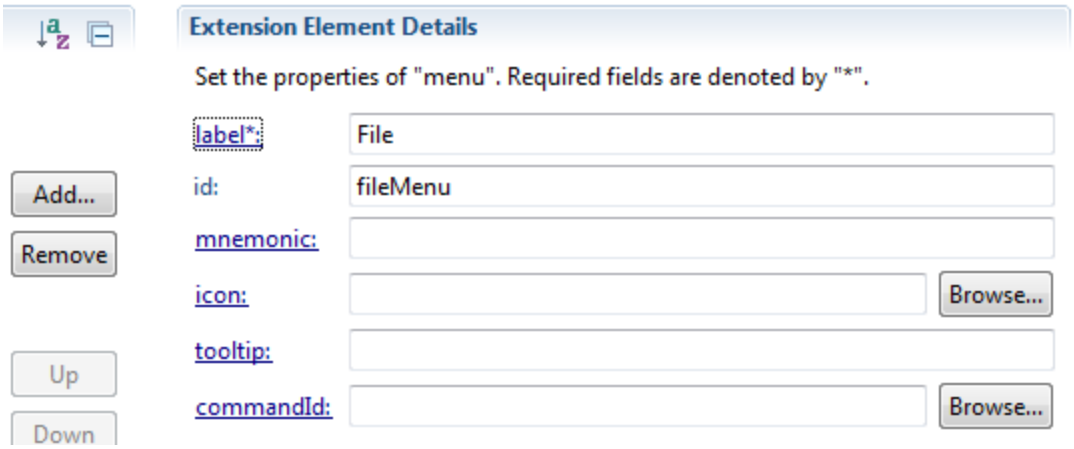


Full in ‘locationURL*’ with ‘menu:org.eclipse.ui.main.menu’

Right click ‘menucontribution’ select ‘New’ -> ‘Menu’

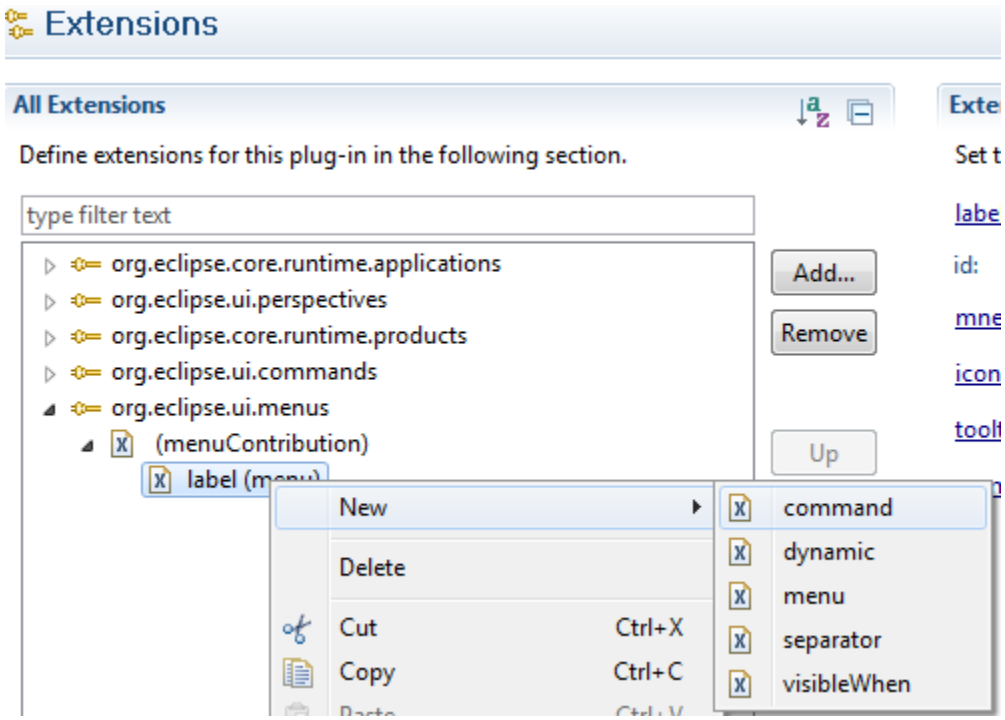


Get

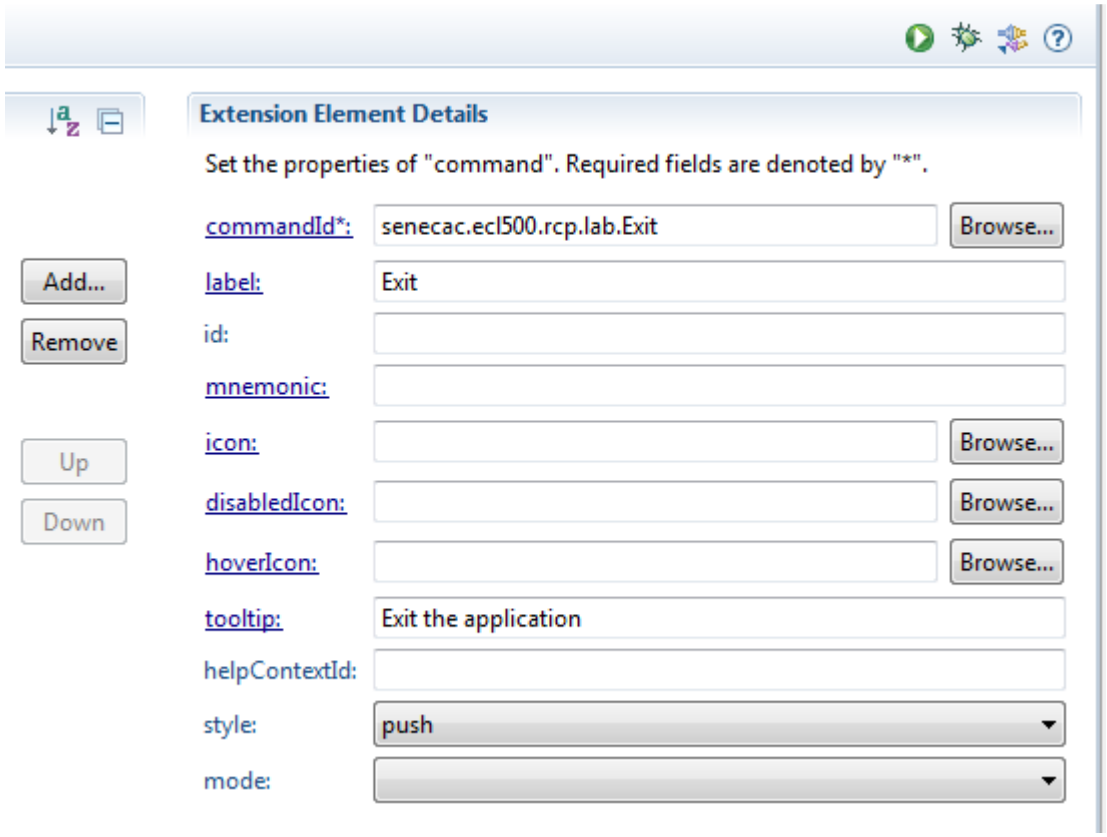


Fill in ‘label*’ with ‘File’ and ‘id’ with ‘fileMenu’

Then right click ‘label(menu)’, ‘New’->’command’



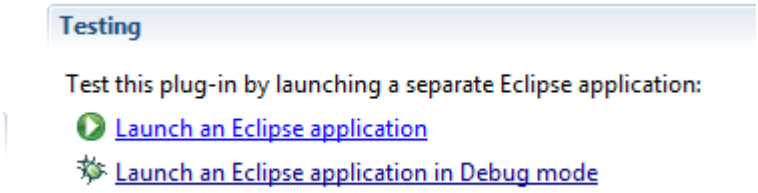
Get



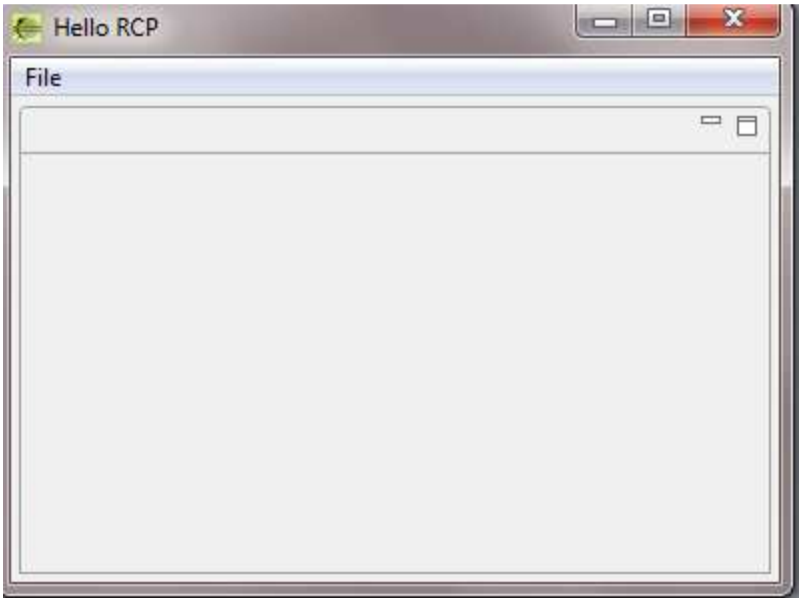
Fill in ‘commandId*’ with ‘senecac.ecl500.rcp.lab.Exit’ ’label’ with ‘Exit ’, and ‘tooltip’ with ‘Exit the application’

Save it.

When click ’Launch an Eclipse application’

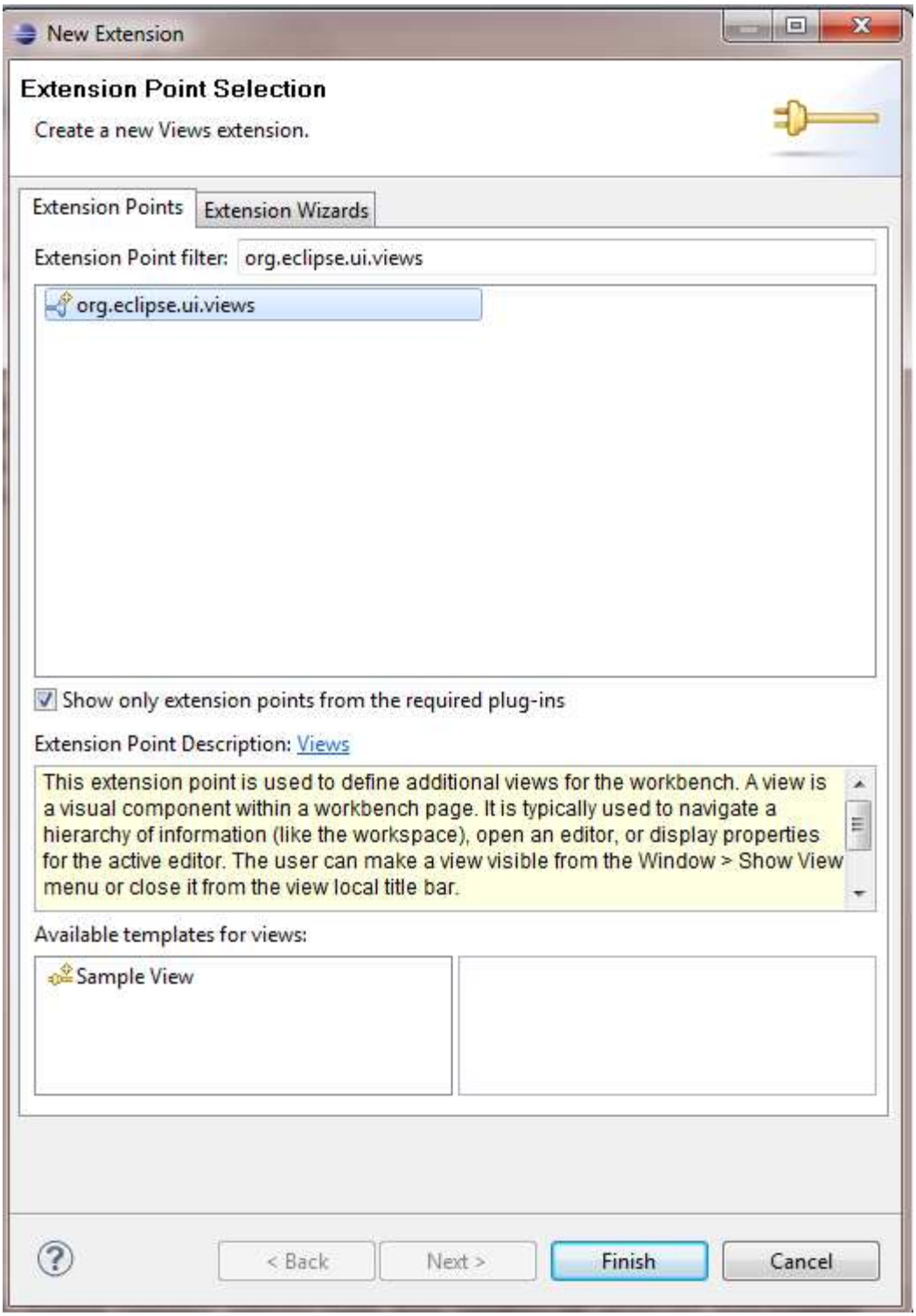


Get the running window

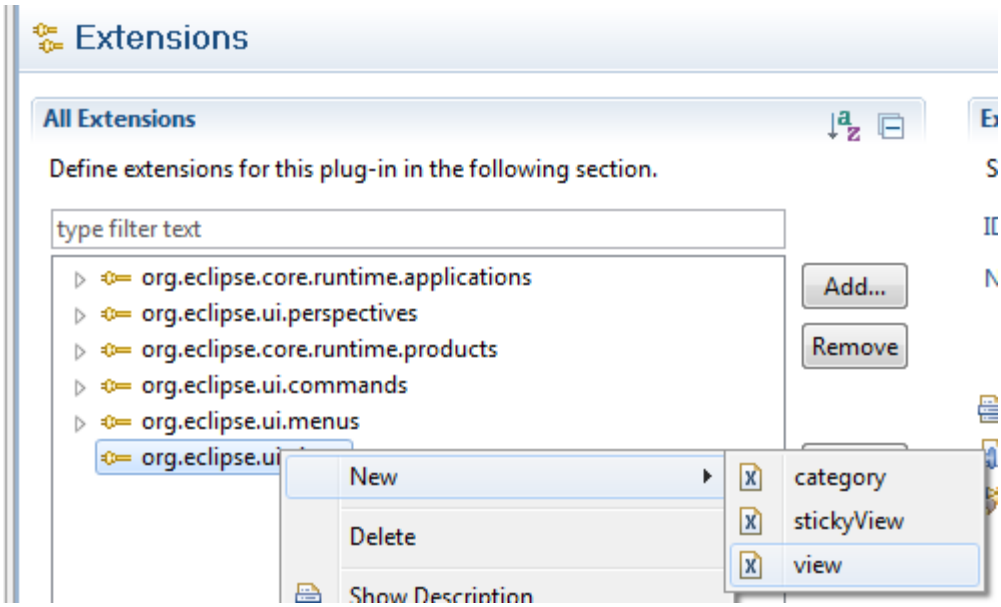


Create view

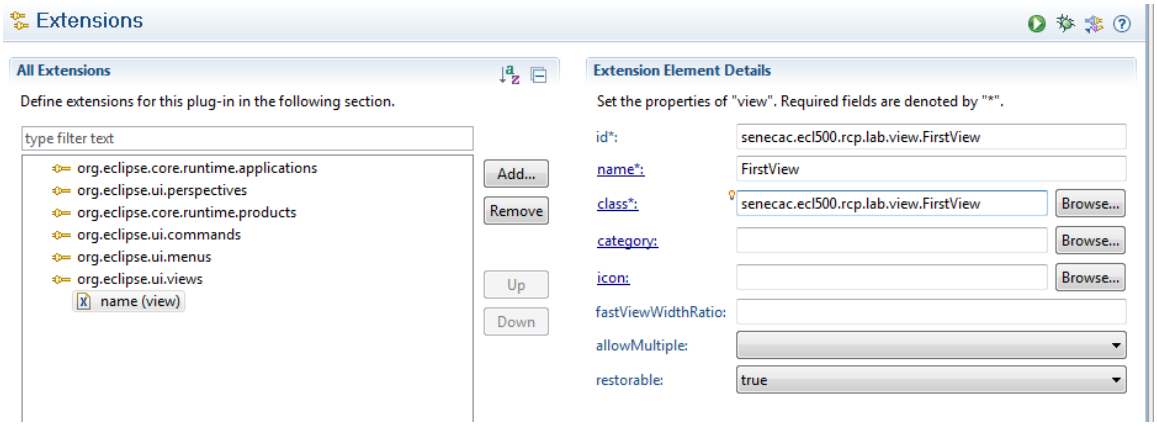
Also at ‘Extensions’ tag, ‘Add’-> ‘org.eclipse.ui.views’



‘Finish’, right click ‘org.eclipse.ui.views’, ‘New’->’view’



Fill in ‘id’ with ‘senecac.ecl500.rcp.lab.view.FirstView’, ‘name’ with ‘FirstView’, and ‘class’ with ‘senecac.ecl500.rcp.lab.view.FirstView’

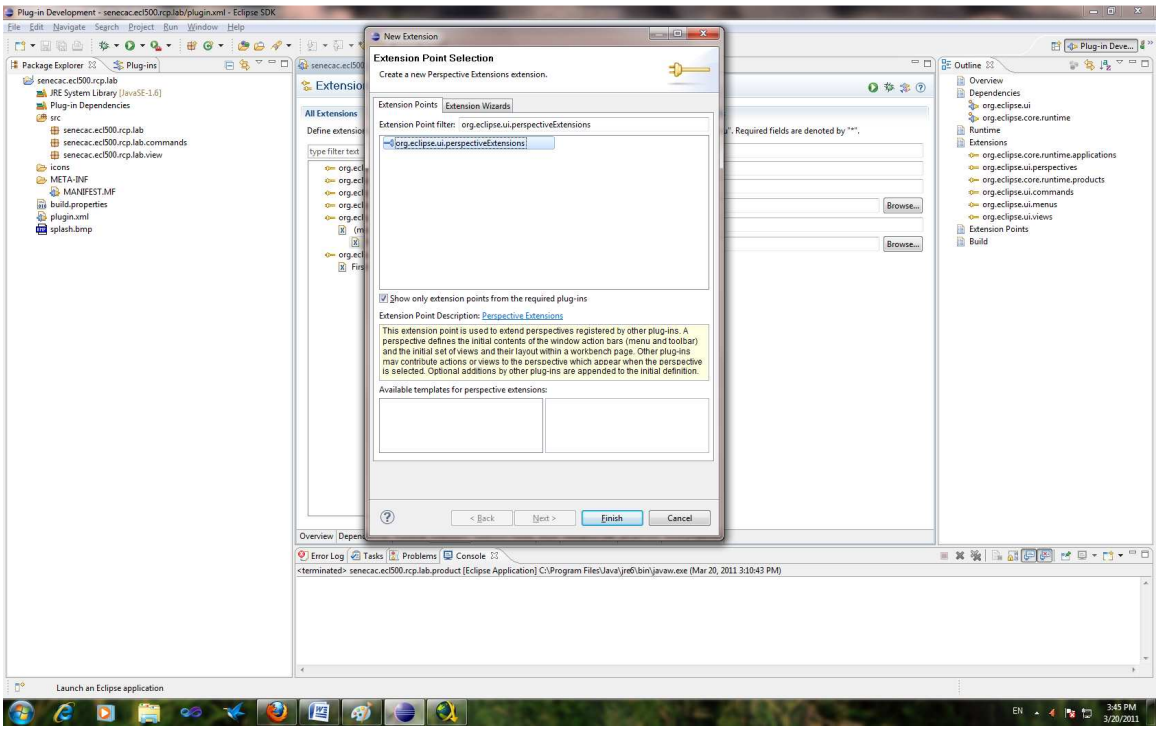


Click hyperlink ‘class’, edit the code with

senecac.ecl500.rcp.lab | FirstView.java

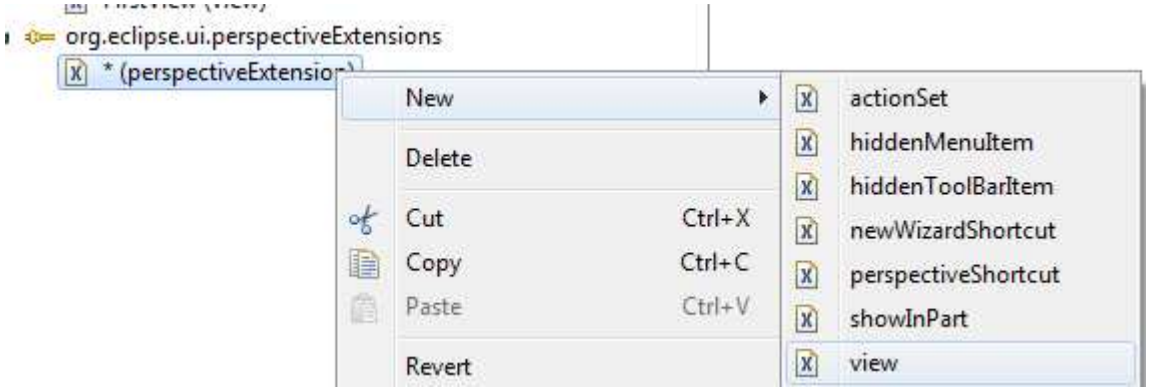
```
package senecac.ecl500.rcp.lab.view;

import org.eclipse.ui.part.ViewPart;

public class FirstView extends ViewPart {

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

    @Override
    public void createPartControl(Composite parent) {
        Text text = new Text(parent, SWT.BORDER);
        text.setText("Imagine a fantastic user interface here");
    }

    @Override
    public void setFocus() {
    }

}
```

Save it, the go back to ‘Extensions’tag, ‘Add’



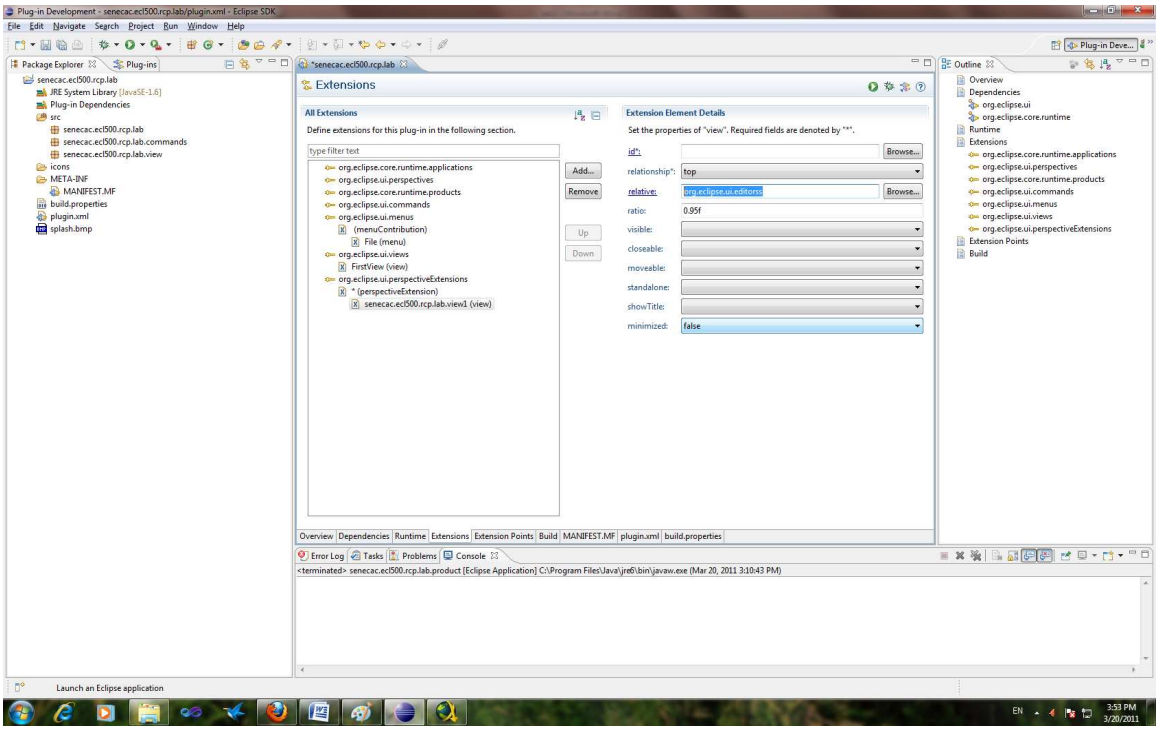
Search package ‘org.eclipse.ui.perspectiveExtensions’, ‘Finish’

Right click "perspectiveExtension"->’New’->’View’

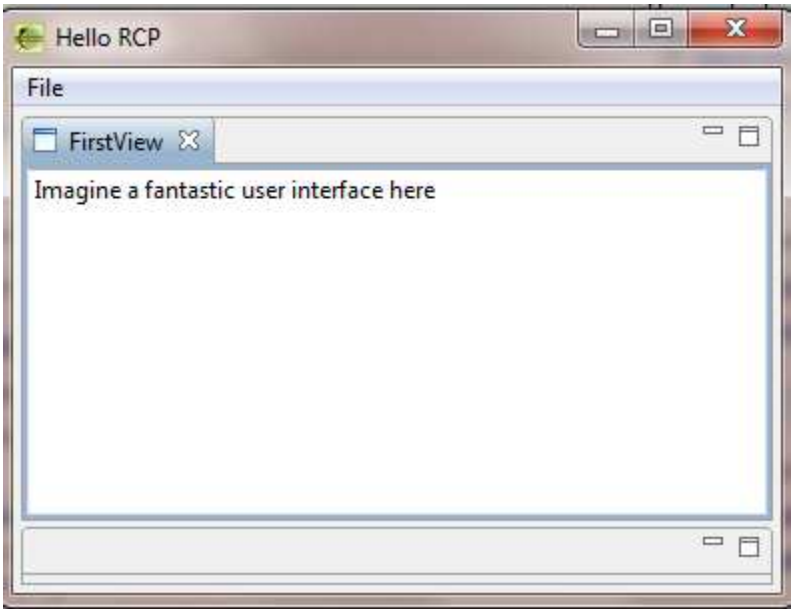


Get

Fill in ‘id’ with ‘senecac.ecl500.rcp.lab.view.FirstView’, ‘relationship’ with ‘top’, ‘relative’ with ‘org.eclipse.ui.editorss, ‘ratio’ with ‘0.95f’, and ‘minimized’ with ‘false’

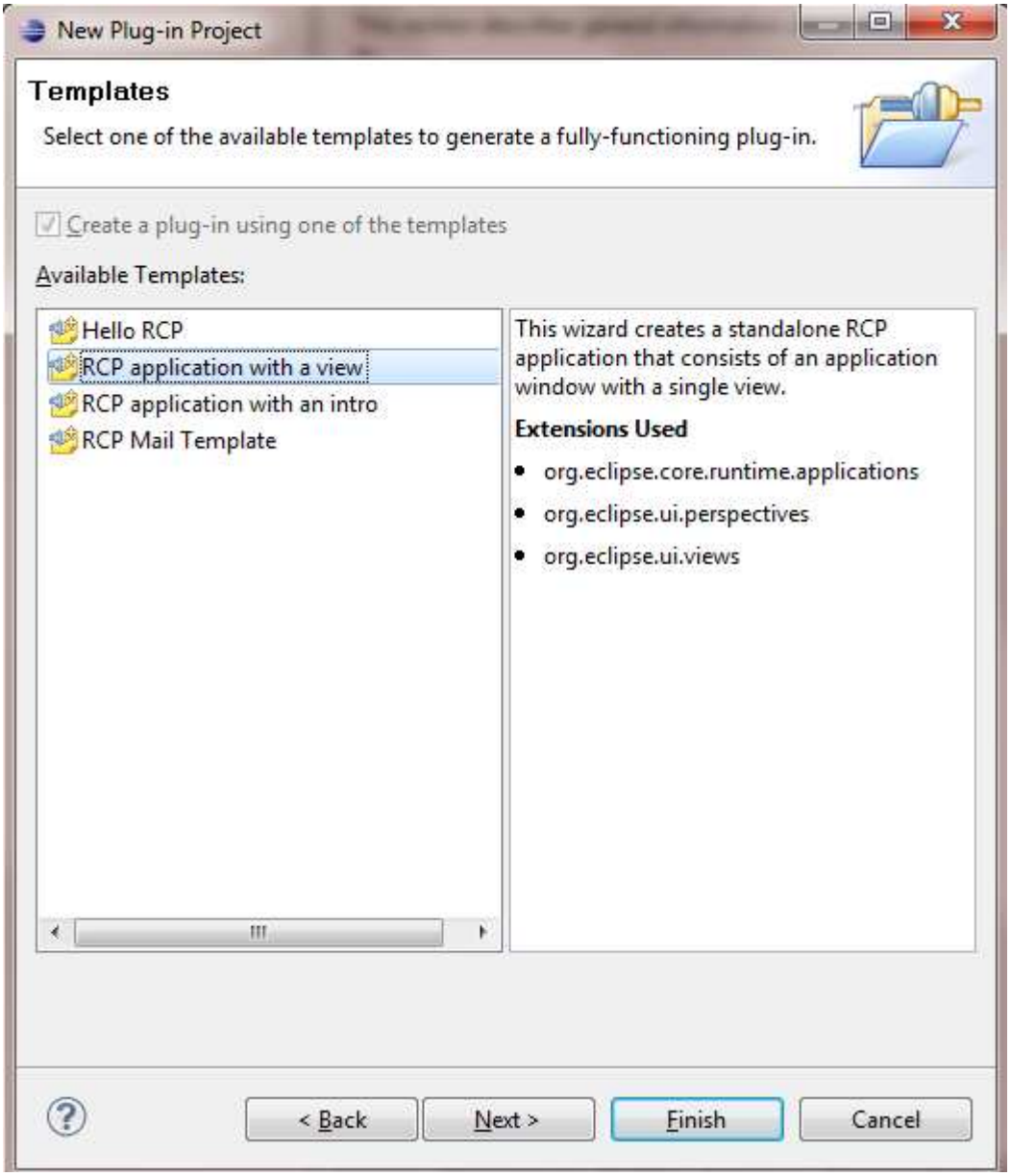


Save and run it again, get



Create editor

Create a new RCP application with a view project named ‘seneca.ecl500.rcp.lab.editor’



'Finish', then create a class 'Person' is package 'seneca.ecl500.rcp.lab.editor.model'

```
package seneca.ecl500.rcp.lab.editor.model;

public class Person {
	private static int counter = 0;
	private int id;
	private String firstName;
	private String lastName;

	public Person(String firstName, String lastName) {
		id = counter++;
		this.firstName = firstName;
```

```
		this.lastName = lastName;
	}

	public int getId() {
		return id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	@Override
	public String toString() {
		return firstName + " " + lastName;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((firstName == null) ? 0 : firstName.hashCode());
		result = prime * result
				+ ((lastName == null) ? 0 : lastName.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
```

```
                return false;
            Person other = (Person) obj;
            if (firstName == null) {
                if (other.firstName != null)
                    return false;
            } else if (!firstName.equals(other.firstName))
                return false;
            if (lastName == null) {
                if (other.lastName != null)
                    return false;
            } else if (!lastName.equals(other.lastName))
                return false;
            return true;
        }

}
```

And create a class ‘MyModel’ in package of ‘seneca.ecl500.rcp.lab.editor.model’

```
package seneca.ecl500.rcp.lab.editor.model;

import java.util.ArrayList;
import java.util.List;

public class MyModel {

    private static MyModel model;
    private List<Person> persons = new ArrayList<Person>();

    private MyModel() {
        Person person = new Person("Hans", "Nase");
        persons.add(person);
        person = new Person("Jim", "Knopf");
        persons.add(person);
    }

    public static MyModel getInstance() {
        if (model == null) {
            model = new MyModel();
        }
        return model;
    }

    public List<Person> getPersons() {
        return persons;
    }

    public Person getPersonById(int id) {
```

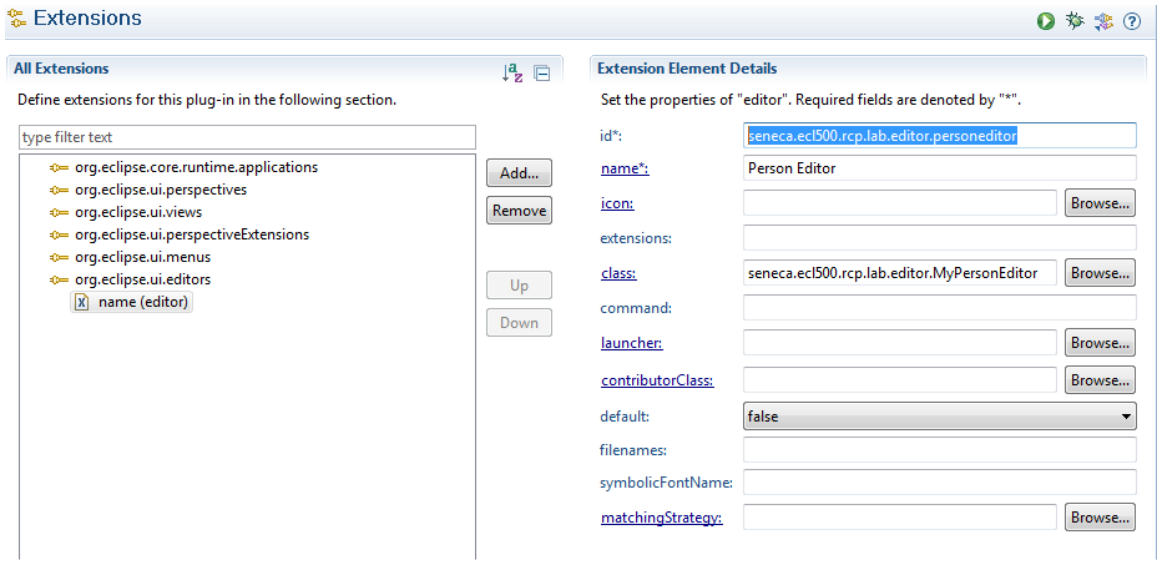
```
                for (Person person : persons) {
                        if (person.getId() == id) {
                                System.out.println("returned");
                                return person;
                        }
                }
                return null;
        }
}
```

Create a class "MyPersonEditorInput" in ‘seneca.ecl500.rcp.lab.editor’ package

```
package seneca.ecl500.rcp.lab.editor;

import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.IPersistableElement;

public class MyPersonEditorInput implements IEditorInput {

   private final int id;

   public MyPersonEditorInput(int id) {
      this.id = id;
   }
   public int getId() {
      return id;
   }

   @Override
   public boolean exists() {
      return true;
   }

   @Override
   public ImageDescriptor getImageDescriptor() {
      return null;
   }

   @Override
   public String getName() {
      return String.valueOf(id);
   }

   @Override
   public IPersistableElement getPersistable() {
      return null;
   }
```

```
    @Override
    public String getToolTipText() {
        return "Displays a person";
    }

    @Override
    public Object getAdapter(Class adapter) {
        return null;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + id;
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        MyPersonEditorInput other = (MyPersonEditorInput) obj;
        if (id != other.id)
            return false;
        return true;
    }


}
```

Open ‘plugin.xml’->go to ’Extensions’->’Add’->’org.eclipse.ui.editors’->’Finish’



Fill in ‘id’ with ‘seneca.ecl500.rcp.lab.editor.personeditor’, ‘name’ with ‘Person Editor’, and ‘class’ with ‘seneca.ecl500.rcp.lab.editor.MyPersonEditor’, then click ‘class’ to create the class of ‘MyPersonEditor’ with the code of

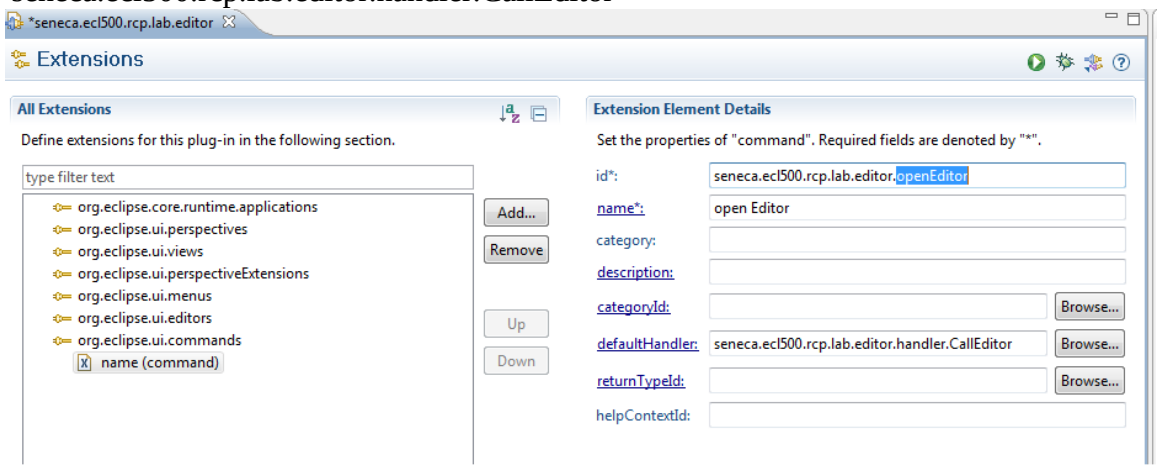
```
package seneca.ecl500.rcp.lab.editor;

import org.eclipse.core.runtime.IProgressMonitor;
import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;
import org.eclipse.ui.IEditorInput;
import org.eclipse.ui.IEditorSite;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.part.EditorPart;

import seneca.ecl500.rcp.lab.editor.model.MyModel;
import seneca.ecl500.rcp.lab.editor.model.Person;

public class MyPersonEditor extends EditorPart {
	public static final String ID = "de.vogella.rcp.editor.example.editor.personeditor";
	private Person person;
	private MyPersonEditorInput input;

	// Will be called before createPartControl
	@Override
	public void init(IEditorSite site, IEditorInput input)
			throws PartInitException {
		if (!(input instanceof MyPersonEditorInput)) {
```

```
                throw new RuntimeException("Wrong input");
            }

            MyPersonEditorInput new_name = (MyPersonEditorInput) input;
            this.input = (MyPersonEditorInput) input;
            setSite(site);
            setInput(input);
            person = MyModel.getInstance().getPersonById(this.input.getId());
            setPartName("Person " + person.getLastName());
        }

        @Override
        public void createPartControl(Composite parent) {
            GridLayout layout = new GridLayout();
            layout.numColumns = 2;
            parent.setLayout(layout);
            Label label1 = new Label(parent, SWT.BORDER);
            label1.setText("First Name");
            Text text = new Text(parent, SWT.BORDER);
            text.setText(person.getFirstName());
            text.setLayoutData(new GridData(SWT.FILL, SWT.BEGINNING, true,
false));

        }

        @Override
        public void doSave(IProgressMonitor monitor) {
            // person.getAddress().setCountry(text2.getText());
        }

        @Override
        public void doSaveAs() {
        }

        @Override
        public boolean isDirty() {
            return false;
        }

        @Override
        public boolean isSaveAsAllowed() {
            return false;
        }

        @Override
        public void setFocus() {
```

}

}

Create command id of ‘seneca.ecl500.rcp.lab.editor.openEditor’ it’s Handler class is ‘seneca.ecl500.rcp.lab.editor.handler.CallEditor’



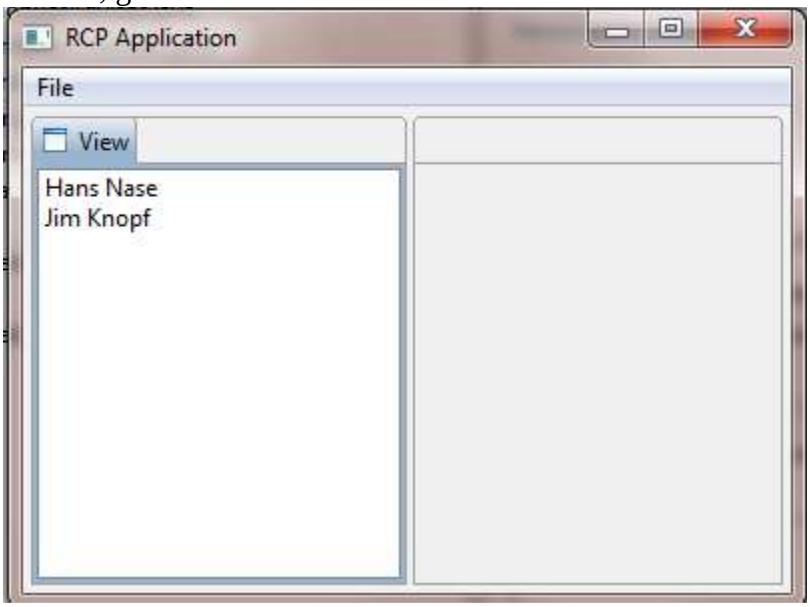
Click ‘defaultHandler’, edit the class with code

```
package seneca.ecl500.rcp.lab.editor.handler;

import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.handlers.HandlerUtil;
import seneca.ecl500.rcp.lab.editor.*;
import seneca.ecl500.rcp.lab.editor.model.*;

public class CallEditor extends AbstractHandler {

	@Override
	public Object execute(ExecutionEvent event) throws ExecutionException {
		System.out.println("called");
		// Get the view
		IWorkbenchWindow window =
HandlerUtil.getActiveWorkbenchWindow(event);
		IWorkbenchPage page = window.getActivePage();
		View view = (View) page.findView(View.ID);
		// Get the selection
		ISelection selection = view.getSite().getSelectionProvider()
				.getSelection();
		if (selection != null && selection instanceof IStructuredSelection) {
```

```
                        Object obj = ((IStructuredSelection) selection).getFirstElement();
                        // If we had a selection lets open the editor
                        if (obj != null) {
                                Person person = (Person) obj;
                                MyPersonEditorInput input = new MyPersonEditorInput(
                                                person.getId());
                                try {
                                        page.openEditor(input, MyPersonEditor.ID);

                                } catch (PartInitException e) {
                                        throw new RuntimeException(e);
                                }
                        }
                }
                return null;
        }

}
```

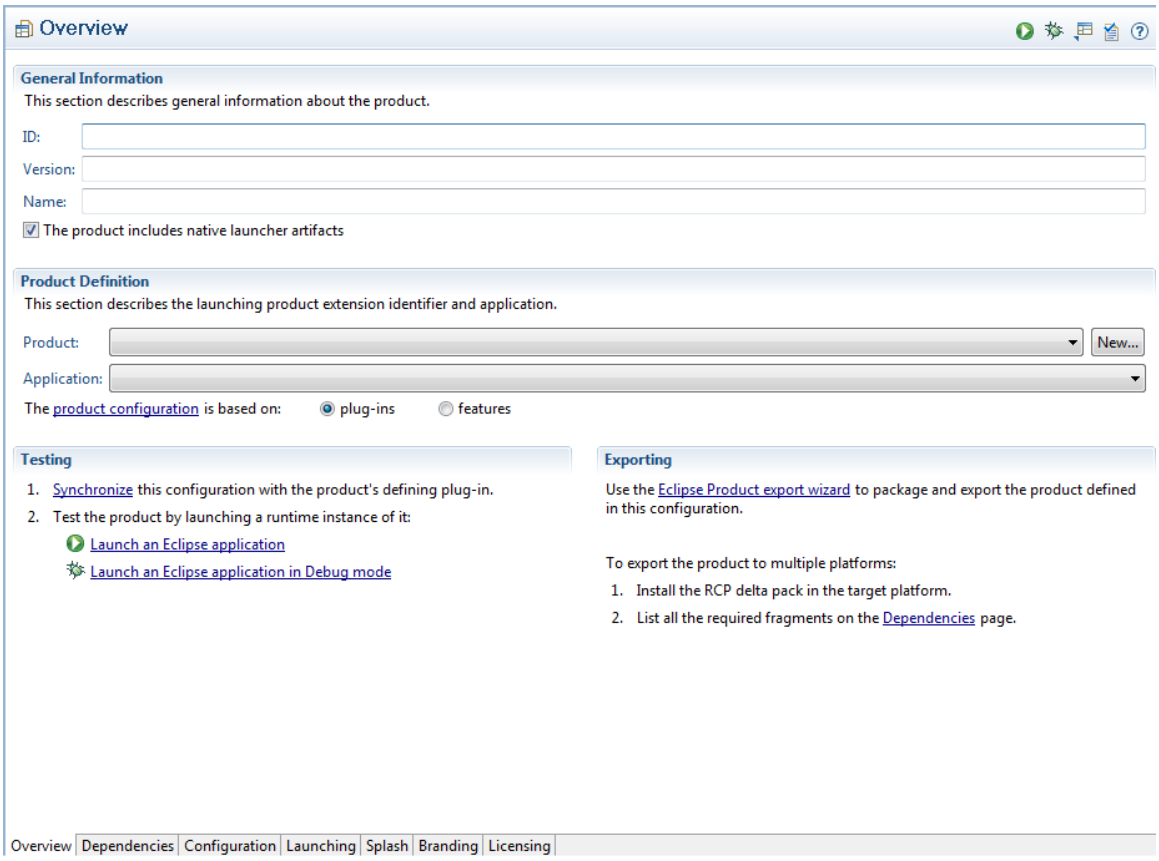
Change the View class to

The run, get

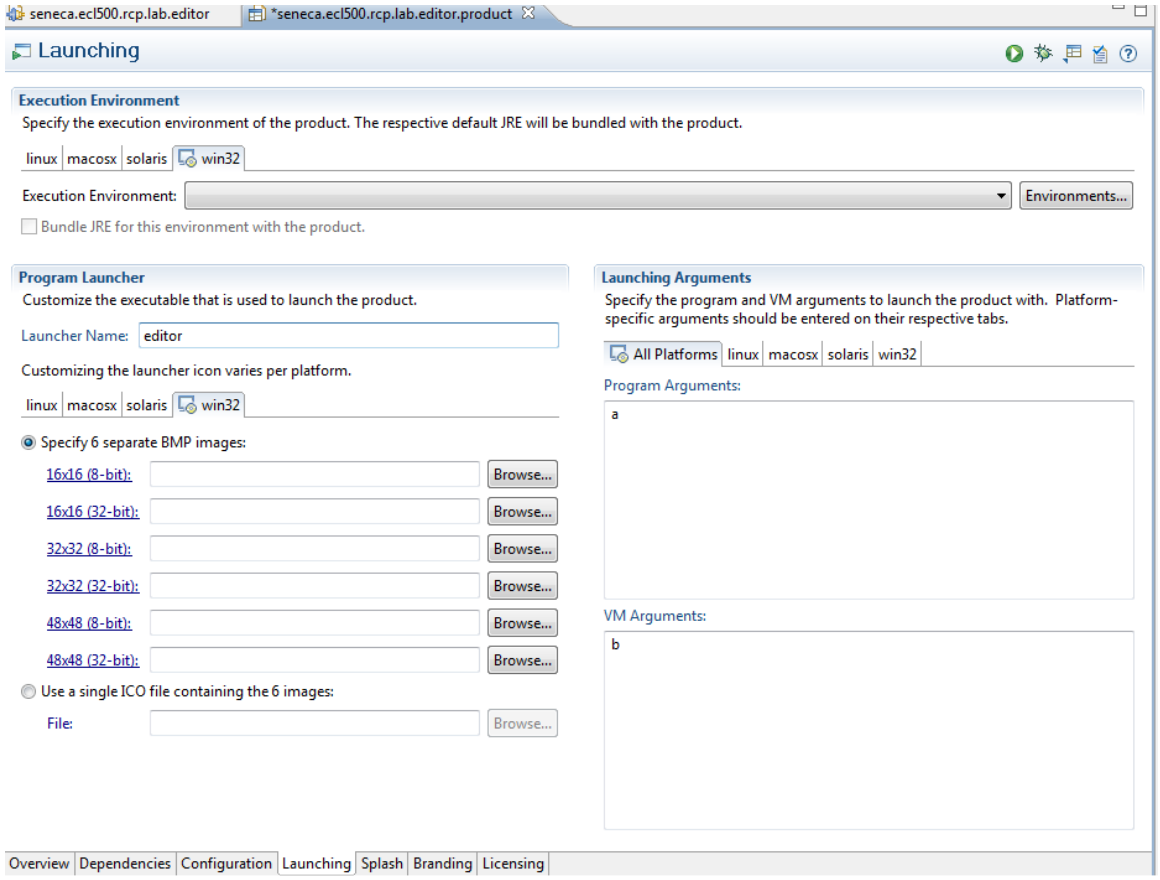


Create product configuration

Right click project name->'New'->'Product configuration', full 'name' with 'seneca.ecl500.rcp.lab.editor.product'->'Finish'



Go to ‘Launching’ tag



Can input ‘Launcher Name’, ‘Program Arguments’, and ‘VM Arguments’