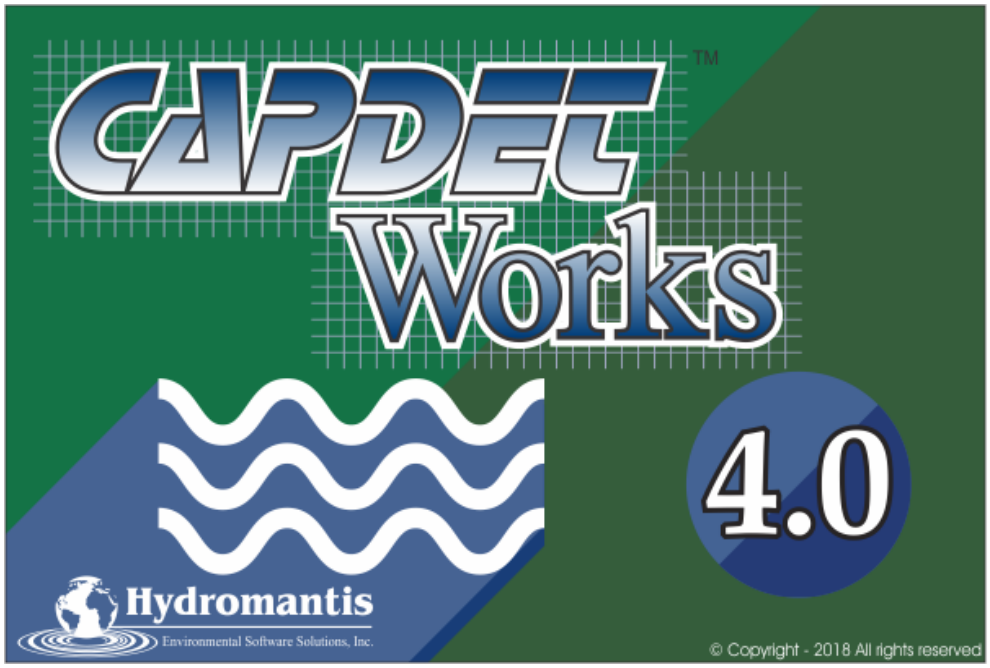
CapdetWorks V4.0

State-of-the-art Software for the Design and Cost Estimation of Wastewater Treatment Plants



Define User Processes

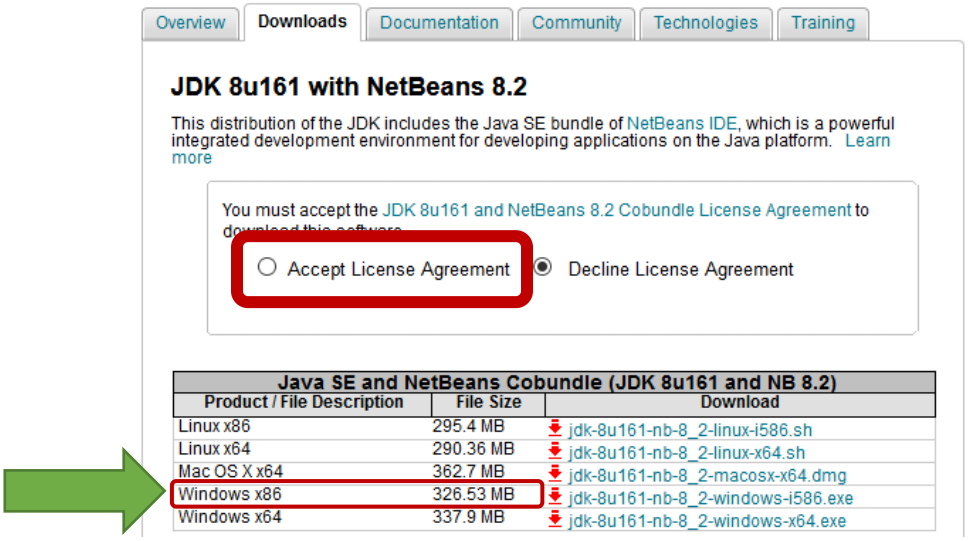
Setting up a Project for Customized Unit Processes in CapdetWorks

We suggest using an IDE (Integrated Development Environment) to ease the set up of the project and writing the code for your customized unit processes. Below is a description of how to download and use the free Netbeans IDE.

1. Download the Netbeans IDE (bundled with the java development kit) from http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html

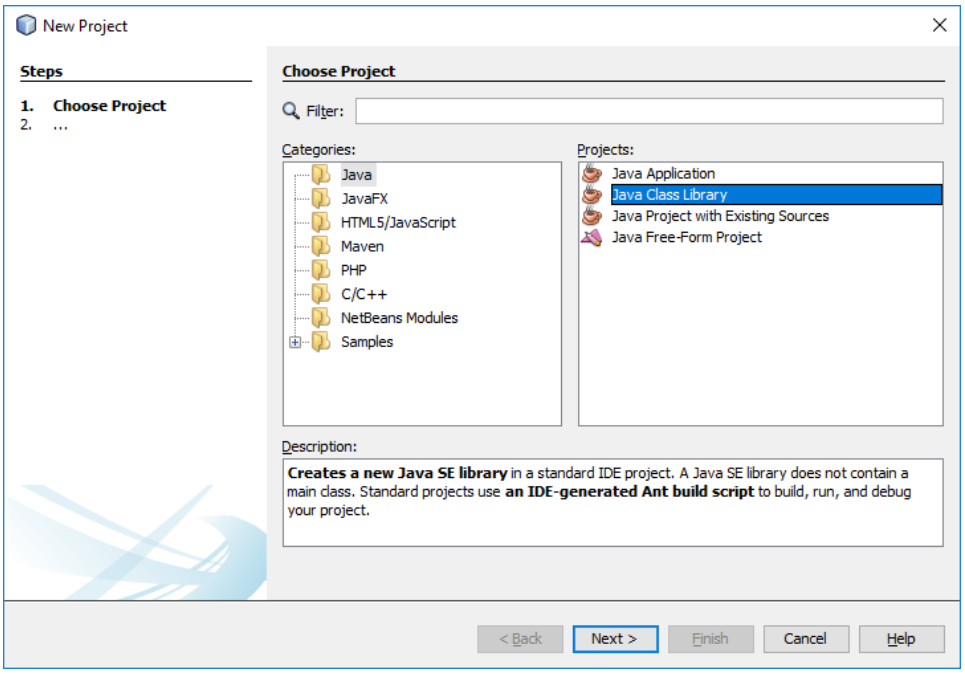
   First, accept the license agreement.
   Then make sure that you download the "Windows x86" version.

   Note: Even if you have a 64 bit system, CapdetWorks itself is only a 32 bit program so you can't develop your code in the 64 bit version. The "Windows x86" 32 bit version runs fine on a 64 bit computer.

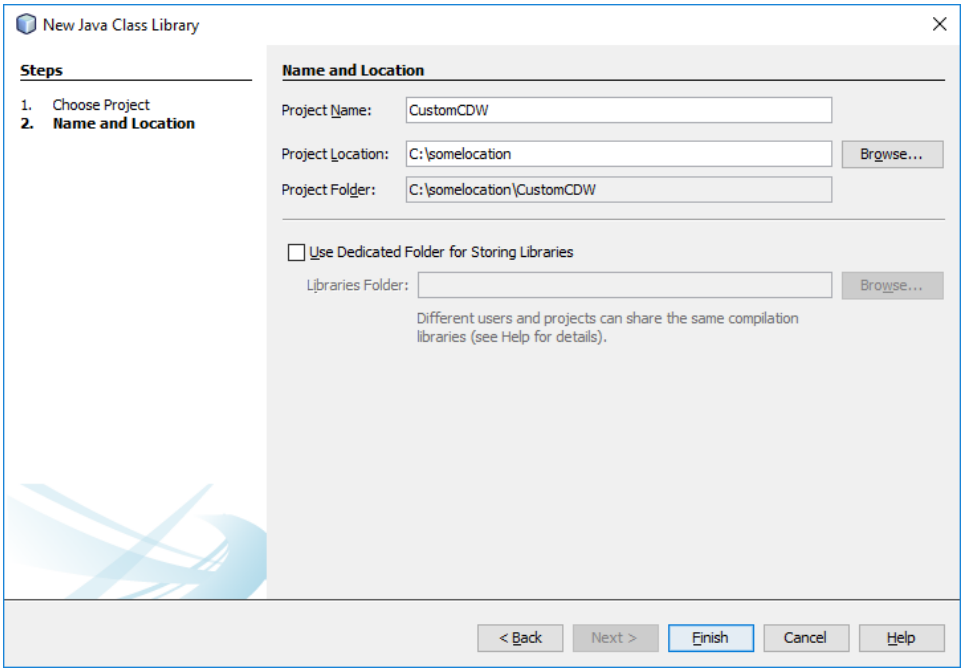


2. Install the program and then start it up.

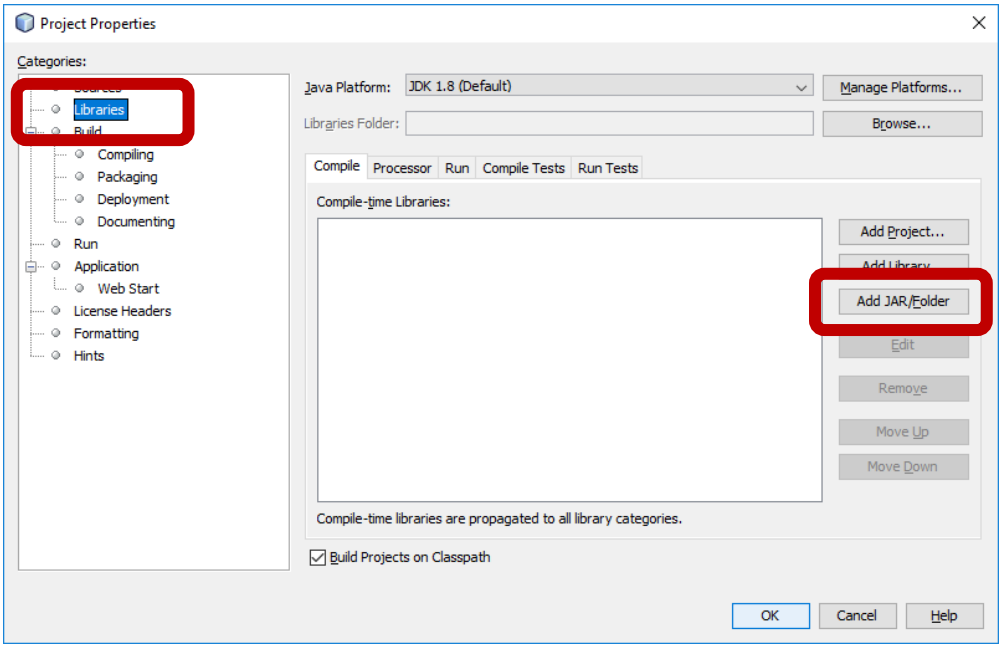
3. From the main menu, go to File -> New Project
   Select “Java Class Library” and hit next.



4. Give your project a name and select a location to save it on your computer. Then click “Finish” to create the project.



5. Go to File -> Project Properties.
   Under the "Libraries" category, click the "Add JAR/Folder" button.

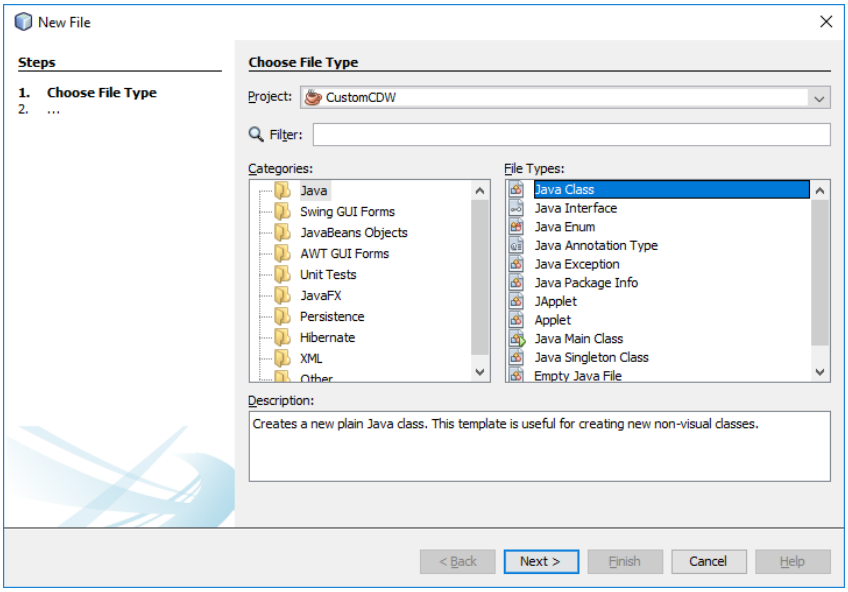


6. Browse to the installation directory of CapdetWorks. In the **'bin'** subdirectory, select the ***cdw.jar*** file and click "Open".

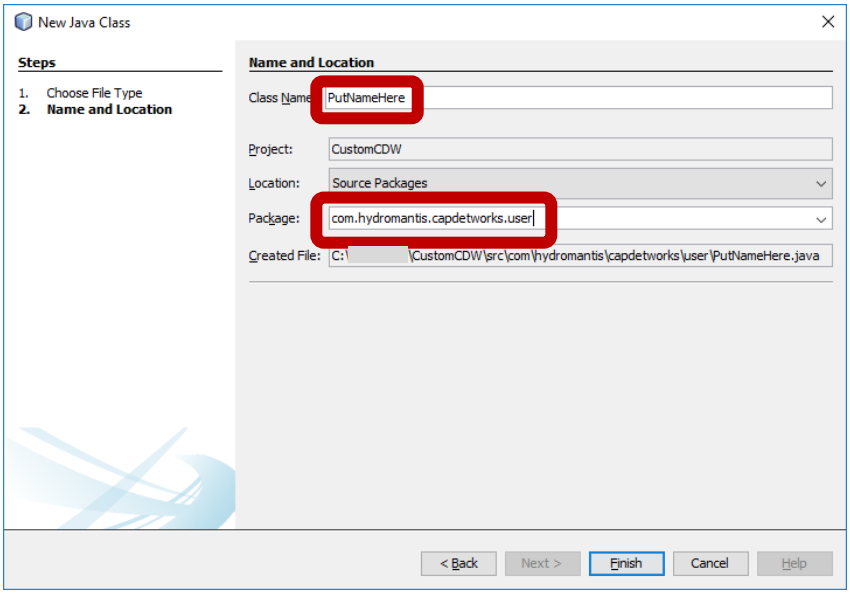
The project has now been set up and we are ready to create the file that will contain the customized code.

Create a file for a customized unit process

1. In Netbeans, go to File -> New File.
   Make sure your project is selected at the top of the dialog.
   Select “Java” from the Categories list.
   Select “Java Class” from the File Types list.
   Click “Next”.



2. Enter a short name for the unit process in the ‘Class Name’ field (no spaces)
   Enter “com.hydromantis.capdetworks.user” as the ‘package’.
   Click ‘Finish’.

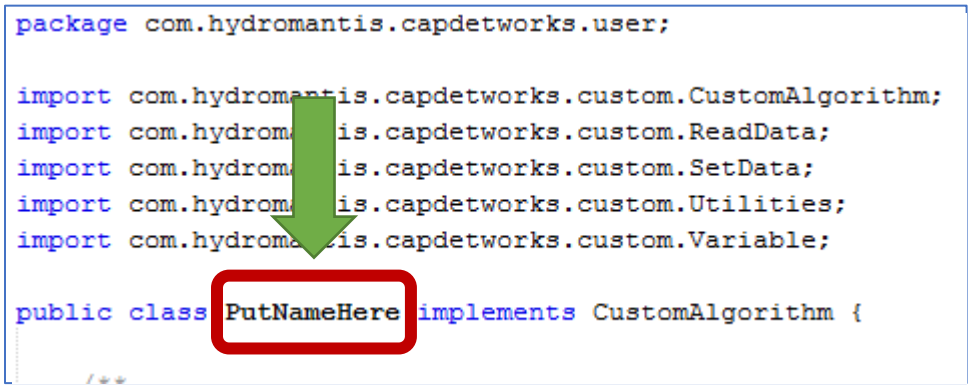


This has now created a file called “PutNameHere.java” (or whatever you chose to name your unit process above) in the appropriate location in your project directory. The file should be open in the Netbeans IDE where you can edit it and the different keywords are coloured differently and it will indicate if you’ve made any syntax errors in the code that you write.

Set up the file with Template Code

You’ve been supplied with a template file (see **Appendix A – Template Code.** The file itself is also available through the Help menu in CapdetWorks) that shows the methods/functions that you must implement and some sample code that you can use to get and set values in CapdetWorks. Open that file in Notepad or some other simple text editor and copy/paste the whole contents into the file you created in Netbeans above.

Rename the class from “PutNameHere” to whatever you named your java class. The class name must correspond to the file name.

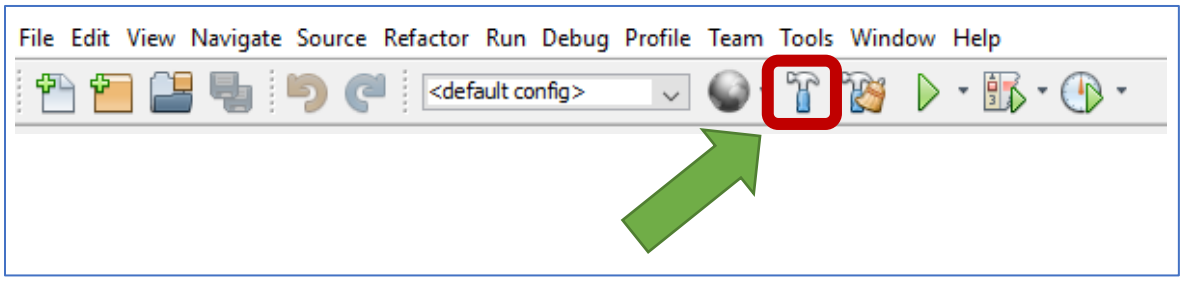


The template file has a lot of comments in it to describe the different sections and methods and what they do. The main method where you’ll write your algorithm is the ‘calculate’ method but the other methods are used to set up your input variables and how they are displayed in the CapdetWorks interface.

At this point, just leave the template code as it is and follow the following instructions to compile the existing code and add it to CapdetWorks. This is just to confirm that you understand how to do this process before trying to write your own code.

Compile Your Code

To compile your code, simply click the 'hammer' button on the Netbeans toolbar.



To confirm that it compiles properly, view the "Output" window (accessed through Windows->Output, if it isn't automatically shown). There should be a "Build Successful" message there or possibly some error messages if you've made some syntax errors in your code.

This will create a 'jar' file in the 'dist' subdirectory wherever you created your project (see "Step 4" of **Setting up a Project for Customized Unit Processes in CapdetWorks**)

Include your Jar File in CapdetWorks

To inform CapdetWorks where it can find your custom unit processes, create a plain text file in the root of the CapdetWorks installation directory called 'userprocesses.txt'.

In that file, create a single line with the fully qualified path to your jar file.

For example:

C:\pathtoproject\dist\CustomCDW.jar

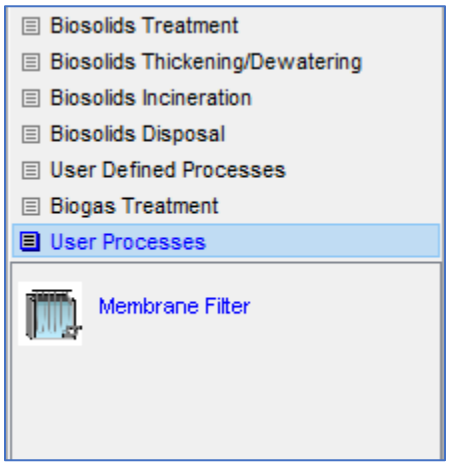
That's it!

When you run CapdetWorks, there should be another process group at the bottom of the process table toolbar where all your custom user processes are located. Drag them out to the drawing board, connect them up and solve the layout to see the results.

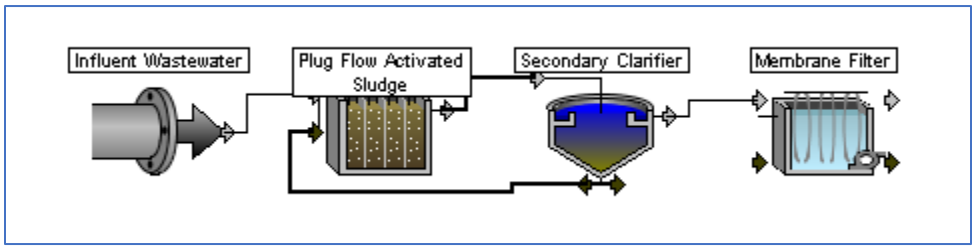
NOTE: Whenever you make changes and recompile your custom code, you'll need to restart CapdetWorks to see the changes.

Results

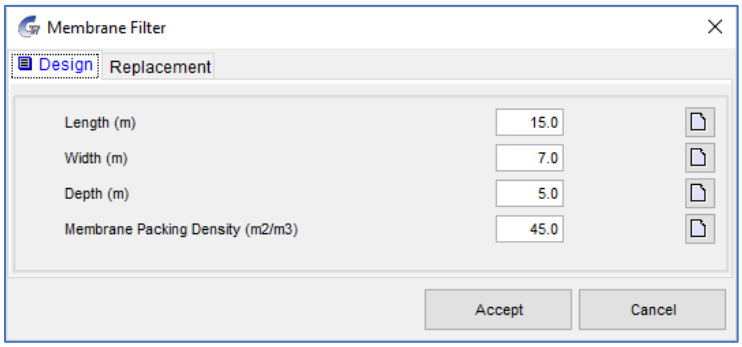
You can find an example of a user-defined Membrane Filter (see **Appendix B – Membrane Filter Example**) along with the template code. After you have done the previous steps, you should be able to see it listed under the User Processes at the bottom of the process table toolbar.

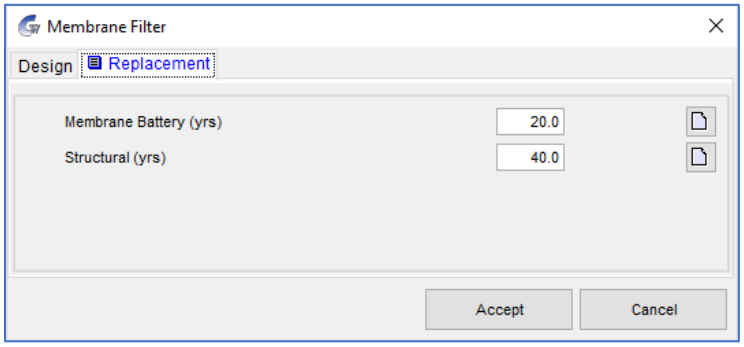


Drag an Influent Wastewater object, a Plug Flow Activated Sludge object and a Membrane Filter object to the drawing board. Connect them as shown in the figure below.

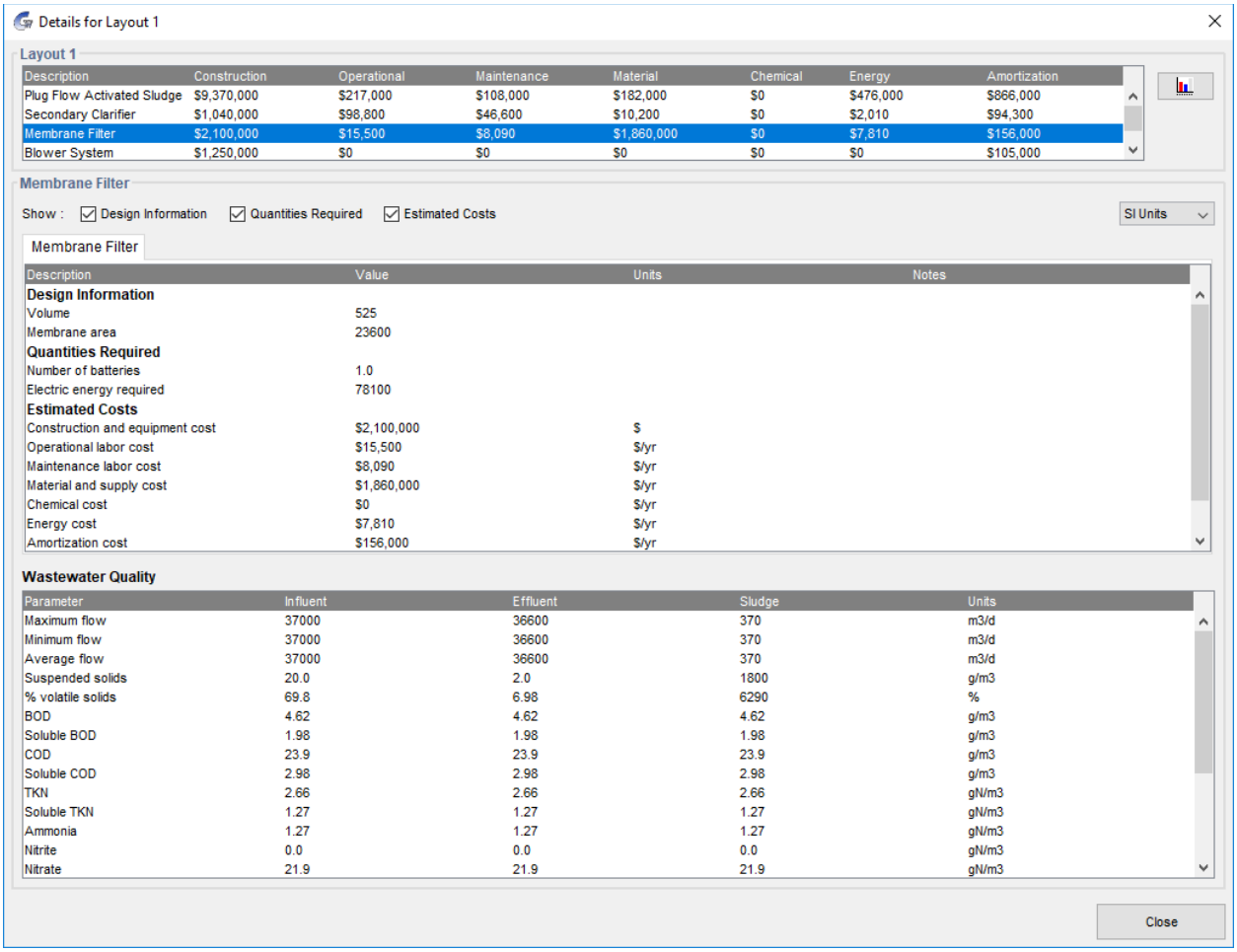


Right-click on the Membrane Filter object and select “Edit Membrane Filter Parameters” option. You can change the parameters or leave them as they are.





Cost the current layout and check the details of cost estimate.



Note: This is just a simple example to demonstrate how to use the template code to generate a user-defined process. The result is not an accurate reflection of the actual costing of a membrane filter.

Appendix A – Template Code

```
package com.hydromantis.capdetworks.user;

import com.hydromantis.capdetworks.custom.CustomAlgorithm;
import com.hydromantis.capdetworks.custom.ReadData;
import com.hydromantis.capdetworks.custom.SetData;
import com.hydromantis.capdetworks.custom.Utilities;
import com.hydromantis.capdetworks.custom.Variable;

public class PutNameHere implements CustomAlgorithm {

    /**
     *
     * @return The label (ie. name) of this unit process
     */
    @Override
    public String getLabel() {
        return "text label";
    }

    /**
     * @return The number of tabs in the process parameter menu
     */
    @Override
    public int getNumberOfInputTabs() {
        return 2;
    }

    /**
     * @return The name of each tab.
     * Note that indices start at zero and go up to (but don't include) the
     * value returned by the getNumberOfInputTabs method
     */
    @Override
    public String getTabName(int tabIndex) {
        if (tabIndex == 0) {
            return "tab1";
        }
        else {
            return "tab2";
        }
    }

    /**
     * The variable object takes 3 arguments. The first is a 'cryptic' name
that
     * is used in the calculate method (see further down this file) to query
the
     * value. The second is the label to display to the user in the input
     * dialog within CapdetWorks. The third is the default value of the
     * variable.
     *
```

```
     * @param tabIndex The index of the tab that the variable should appear
on.
     * Note that indices start at zero and go up to (but don't include) the
     * value returned by the getNumberOfInputTabs method
     * @return The variables on that requested tab
     */
    @Override
    public Variable[] getInputs(int tabIndex) {
        if (tabIndex == 0) {
            return new Variable[]{
                new Variable("cryptic1", "desc1", 1.0),
                new Variable("cryptic2", "desc2", 2.0),
                new Variable("cryptic3", "desc3", 3.0),
                new Variable("cryptic4", "desc4", 4.0),
                new Variable("cryptic5", "desc5", 5.0)
            };
        }
        else {
            return new Variable[]{
                new Variable("cryptic11", "desc11", 1.0),
                new Variable("cryptic12", "desc12", 2.0),
                new Variable("cryptic13", "desc13", 3.0),
                new Variable("cryptic14", "desc14", 4.0),
                new Variable("cryptic15", "desc15", 5.0)
            };
        }
    }

    /**
     * @return Can be either an absolute path OR relative to the user's jar
     * location (if the file doesn't exist, a simple box will be used)
     */
    @Override
    public String getImageFile() {
        return "c:\\test-image.png";
    }

    /**
     * This is where the custom algorithm is located. You can query values,
use
     * them to calculate something and then set values
     *
     * @param in The data that you can query.
     * @param out The outgoing stream's data that you can set.
     * @param utils Access to various utility methods.
     */
    @Override
    public void calculate(ReadData in, SetData out, Utilities utils) {
        //read an value from the user input dialog of the process.
        //The string must match one of the variables created above.
        //The returned value can then be used in your calculations.
        double value = in.getInput("cryptic1");

        //examples of querying the incoming stream's data
        //obviously use better variable names than 'streamValue1' etc
        double streamValue1 = in.getStreamValue(MAX_FLOW);
        double streamValue2 = in.getStreamValue(MIN_FLOW);
```

```
        double streamValue3 = in.getStreamValue(AVE_FLOW);
        double streamValue4 = in.getStreamValue(TSS);
        double streamValue5 = in.getStreamValue(VS);
        double streamValue6 = in.getStreamValue(BOD);
        double streamValue7 = in.getStreamValue(SBOD);
        double streamValue8 = in.getStreamValue(COD);
        double streamValue9 = in.getStreamValue(SCOD);
        double streamValue10 = in.getStreamValue(TKN);
        double streamValue11 = in.getStreamValue(STKN);
        double streamValue12 = in.getStreamValue(NH3);
        double streamValue13 = in.getStreamValue(NO2);
        double streamValue14 = in.getStreamValue(NO3);
        double streamValue15 = in.getStreamValue(TP);
        double streamValue16 = in.getStreamValue(PH);
        double streamValue17 = in.getStreamValue(CATIONS);
        double streamValue18 = in.getStreamValue(ANIONS);
        double streamValue19 = in.getStreamValue(SS);
        double streamValue20 = in.getStreamValue(OIL);
        double streamValue21 = in.getStreamValue(S_TEMP);
        double streamValue22 = in.getStreamValue(W_TEMP);

        //read a value from the standard equipment data (that has been
modified using the cost indices, if applicable)
        double costtest = in.getStandardEquipmentCost("CCHILL");

        //read a value from the user's custom equipment data (that has been
modified using the cost indices, if applicable)
        double costtest2 = in.getCustomEquipmentCost("testItem");


        //showing the results of various calculations in the
Design/Quantities/Notes section in the output window of CapdetWorks
        out.addDesignValue("design output", 87878787.9);
        out.addQuantitiesValue("quantity value", costtest);
        out.addNote("Just some text for the notes section.");

        //Set the stream state values.  This example just shows the
min/max/ave flows but you can set all the stream states shown above in the
querying example.
        //Note: By default the regular wastewater stream is a copy of the
incoming stream.
        //      You can either set ALL of the stream state values
accordingly, or only change the ones that are different from the incoming
stream.
        //      By default the sludge stream is empty.  If you'd like to use
it, you need to set ALL of the stream states accordingly.
        double outFlow = streamValue3/2.0;
        out.setStreamValue(MAX_FLOW, outFlow);
        out.setStreamValue(MIN_FLOW, outFlow);
        out.setStreamValue(AVE_FLOW, outFlow);
        out.setSludgeStreamValue(MAX_FLOW, outFlow);
        out.setSludgeStreamValue(MIN_FLOW, outFlow);
        out.setSludgeStreamValue(AVE_FLOW, outFlow);

        //If you want to use the amortization calculation in CapdetWorks,
this is an example.
```

```
        //The first argument is the principal amount ($).  The second is the
replacement schedule (in years).
        double amort = utils.calculateAmortization(1000.0, 20);

        //Set the costs for the unit process
        out.setConstructionCost(1234.0); // $
        out.setOperationLaborHours(5.0); // hours (the cost is calculated
from the hourly labour cost in CapdetWorks)
        out.setMaintenanceLaborHours(5.0); // hours (the cost is calculated
from the hourly labour cost in CapdetWorks)
        out.setMaterialCost(1234.0);
        out.setChemicalCost(1234.0);
        out.setEnergyCost(1234.0);
        out.setAmortizationCost(amort);

        //set the amount of land that is required for this process (in acres)
        out.setLandArea(500.0);

    }

}
```

Appendix B – Membrane Filter Example

This is an example of a customized membrane filter using the template code. Changes made to the template code have been highlighted in yellow.

```
package com.hydromantis.capdetworks.user;

import com.hydromantis.capdetworks.custom.CustomAlgorithm;
import com.hydromantis.capdetworks.custom.ReadData;
import com.hydromantis.capdetworks.custom.SetData;
import com.hydromantis.capdetworks.custom.Utilities;
import com.hydromantis.capdetworks.custom.Variable;

public class memfiler implements CustomAlgorithm {

    /**
     * @return The label (ie. name) of this unit process
     */
    @Override
    public String getLabel() {
        return "Membrane Filter";
    }

    /**
     * @return The number of tabs in the process parameter menu
     */
    @Override
    public int getNumberOfInputTabs() {
        return 2;
    }

    /**
     * @return The name of each tab.
     * Note that indices start at zero and go up to (but don't include) the
     * value returned by the getNumberOfInputTabs method
     */
    @Override
    public String getTabName(int tabIndex) {
        if (tabIndex == 0) {
            return "Design";
        }
        else {
            return "Replacement";
        }
    }

    /**
     * The variable object takes 3 arguments. The first is a 'cryptic' name
that
     * is used in the calculate method (see further down this file) to query
the
     * value. The second is the label to display to the user in the input
     * dialog within CapdetWorks. The third is the default value of the
     * variable.
```

```
     *
     * @param tabIndex The index of the tab that the variable should appear
on.
     * Note that indices start at zero and go up to (but don't include) the
     * value returned by the getNumberOfInputTabs method
     * @return The variables on that requested tab
     */
    @Override
    public Variable[] getInputs(int tabIndex) {
        if (tabIndex == 0) {
            return new Variable[]{
                new Variable("LENGTH", "Length (m)", 15.0),
                new Variable("WIDTH", "Width (m)", 7.0),
                new Variable("DEPTH", "Depth (m)", 5.0),
                new Variable("MPACKD", "Membrane Packing Density (m2/m3)",
45.0),
            };
        }
        else {
            return new Variable[]{
                new Variable("RSSM", "Membrane Battery (yrs)", 20.0),
                new Variable("RSST", "Structural (yrs)", 40.0),
            };
        }
    }

    /**
     * @return Can be either an absolute path OR relative to the user's jar
     * location (if the file doesn't exist, a simple box will be used)
     */
    @Override
    public String getImageFile() {
        return "C:\\Users\\Pictures\\membrane.png";
    }

    /**
     * This is where the custom algorithm is located. You can query values,
use
     * them to calculate something and then set values
     *
     * @param in The data that you can query.
     * @param out The outgoing stream's data that you can set.
     * @param utils Access to various utility methods.
     */
    @Override
    public void calculate(ReadData in, SetData out, Utilities utils) {
        //read an value from the user input dialog of the process.
        //The string must match one of the variables created above.
        //The returned value can then be used in your calculations.
        double LENGTH = in.getInput("LENGTH");
        double WIDTH = in.getInput("WIDTH");
        double DEPTH = in.getInput("DEPTH");
        double MPACKD = in.getInput("MPACKD");
        double RSST = in.getInput("RSST");

        //examples of querying the incoming stream's data
        double inFlow = in.getStreamValue(AVE_FLOW);
```

```
        double inTSS = in.getStreamValue(TSS);
        double inVS = in.getStreamValue(VS);
        double inBOD = in.getStreamValue(BOD);
        double inSBOD = in.getStreamValue(SBOD);
        double inCOD = in.getStreamValue(COD);
        double inSCOD = in.getStreamValue(SCOD);
        double inTKN = in.getStreamValue(TKN);
        double inSTKN = in.getStreamValue(STKN);
        double inNH3 = in.getStreamValue(NH3);
        double inNO2 = in.getStreamValue(NO2);
        double inNO3 = in.getStreamValue(NO3);
        double inTP = in.getStreamValue(TP);
        double inPH = in.getStreamValue(PH);
        double inCations = in.getStreamValue(CATIONS);
        double inAnions = in.getStreamValue(ANIONS);
        double inSS = in.getStreamValue(SS);
        double inOil = in.getStreamValue(OIL);
        double inS_Temp = in.getStreamValue(S_TEMP);
        double inW_Temp = in.getStreamValue(W_TEMP);

        //read a value from the standard equipment data (that has been
modified using the cost indices, if applicable)
        double costmem = in.getStandardEquipmentCost("COSTMEM"); // cost of 1
sqft membrane


        //calculating design variables and stream values
        int numBatteries = (int)(inFlow/100 + 1);
        double Volume = LENGTH * WIDTH * DEPTH;
        double memArea = Volume * MPACKD;
        double energyRequired = 8000 * inFlow; // assumes for every 1 MGD of
flow (default units), it requires 2.5 kWh of electricity
        double sludgeFlow = inFlow*0.01; // assumes 1% of incoming flow
becomes sludge
        double outFlow = inFlow*0.99;
        double outTSS = 0.1*inTSS; // assumes 90% TSS removal efficiency
        double sludgeTSS = (inTSS * inFlow - outTSS * outFlow)/sludgeFlow; //
mass balance on TSS
        double outVS = 0.1*inVS;
        double sludgeVS = (inVS * inFlow - outVS * outFlow)/sludgeFlow;


        //showing the results of various calculations in the
Design/Quantities/Notes section in the output window of CapdetWorks
        out.addDesignValue("Volume", Volume);
        out.addDesignValue("Membrane area", memArea);
        out.addQuantitiesValue("Number of batteries", numBatteries);
        out.addQuantitiesValue("Electric energy required", energyRequired);



        //Set the stream state values.  This example just shows the
min/max/ave flows but you can set all the stream states shown above in the
querying example.
        //Note: By default the regular wastewater stream is a copy of the
incoming stream.
```

```
        //      You can either set ALL of the stream state values
accordingly, or only change the ones that are different from the incoming
stream.
        //      By default the sludge stream is empty.  If you'd like to use
it, you need to set ALL of the stream states accordingly.
        out.setStreamValue(MAX_FLOW, outFlow);
        out.setStreamValue(MIN_FLOW, outFlow);
        out.setStreamValue(AVE_FLOW, outFlow);
        out.setSludgeStreamValue(MAX_FLOW, sludgeFlow);
        out.setSludgeStreamValue(MIN_FLOW, sludgeFlow);
        out.setSludgeStreamValue(AVE_FLOW, sludgeFlow);
        out.setStreamValue(TSS, outTSS);
        out.setSludgeStreamValue(TSS, sludgeTSS);
        out.setStreamValue(VS, outVS);
        out.setSludgeStreamValue(VS, sludgeVS);
        out.setSludgeStreamValue(BOD, inBOD);
        out.setSludgeStreamValue(SBOD, inSBOD);
        out.setSludgeStreamValue(COD, inCOD);
        out.setSludgeStreamValue(SCOD, inSCOD);
        out.setSludgeStreamValue(TKN, inTKN);
        out.setSludgeStreamValue(STKN, inSTKN);
        out.setSludgeStreamValue(NH3, inNH3);
        out.setSludgeStreamValue(NO2, inNO2);
        out.setSludgeStreamValue(NO3, inNO3);
        out.setSludgeStreamValue(TP, inTP);
        out.setSludgeStreamValue(PH, inPH);
        out.setSludgeStreamValue(CATIONS, inCations);
        out.setSludgeStreamValue(ANIONS, inAnions);
        out.setSludgeStreamValue(SS, inSS);
        out.setSludgeStreamValue(OIL, inOil);
        out.setSludgeStreamValue(S_TEMP, inS_Temp);
        out.setSludgeStreamValue(W_TEMP, inW_Temp);

        //If you want to use the amortization calculation in CapdetWorks,
this is an example.
        //The first argument is the principal amount ($).  The second is the
replacement schedule (in years).

        double membraneCost = costmem * memArea * 10.7639; // 10.7639 is the
conversion factor from m2 to sqft
        double amort = utils.calculateAmortization(membraneCost, RSST);
        double constructCost = 4000 * Volume; // assumes for every m3 of
volume, the construction cost is $4000
        double energyCost = energyRequired * 0.1; // assumes $0.1/kWh of
electricity

        //Set the costs for the unit process
        out.setConstructionCost(constructCost); // $
        out.setOperationLaborHours(300.0); // hours (the cost is calculated
from the hourly labour cost in CapdetWorks)
        out.setMaintenanceLaborHours(200.0); // hours (the cost is calculated
from the hourly labour cost in CapdetWorks)
        out.setMaterialCost(membraneCost);
        out.setChemicalCost(0.0);
        out.setEnergyCost(energyCost);
        out.setAmortizationCost(amort);
```

```
        //set the amount of land that is required for this process (in acres)
        out.setLandArea(500.0);

    }

}
```