

Tutorial: Using Phidgets in JCreator

1. Install

Note: Since it is impossible to train all coaches to become expert in all programming tools and all new (beta) versions of phidgets, only the tools mentioned in this document and the versions available on the TU/e website are supported (PC only). You are allowed to use other tools/versions, but by doing this you risk losing Phidget support.

Install the following components:

- Phidget driver including webservice
(http://w3.id.tue.nl/nl/de_faculteit/servicedesk/e_atelier/phidgets/java/): required for windows to use the phidget hardware. The webservice that is included in this package, enable to connect to the phidget components by using a TCP/IP link (either locally or over the internet)

When the phidget webservice starts for the first time, the windows firewall might pop-up a warning message. Select to unblock the programmer.

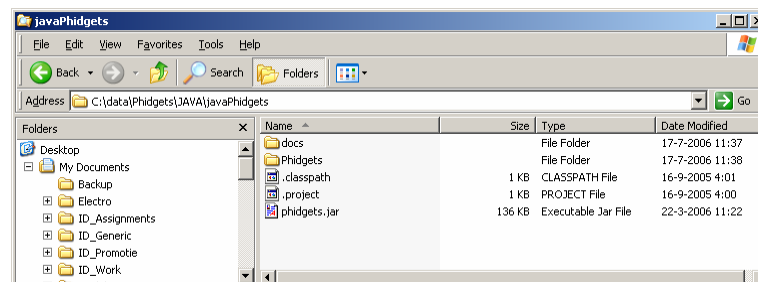


- A JAVA Software Development Kit
Preferred version: SUN Java 2 Platform Standard Edition (J2SE) (www.sun.com)
Note: if you used the JAVA programming language before, you can uninstall a version that was installed earlier on your PC, or just use this old version instead.

- A JAVA programming environment.
The preferred IDE is JCreator (many useful functions, but still fast and easy to use)

After installing, start JCreator and select the directory where you want to store your projects (for example: My Documents\Phidgets\Java): Configure > options > Directories > Default project directory.

- Phidget JAVA libraries
(http://w3.id.tue.nl/nl/de_faculteit/servicedesk/e_atelier/phidgets/java/)
After unpacking this zip-archive, copy the contents to a directory on your PC. For example. My documents\Phidgets\JAVA

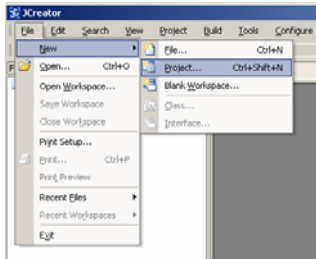


Note: Appendix A shows how Netbeans can be used to create this demo

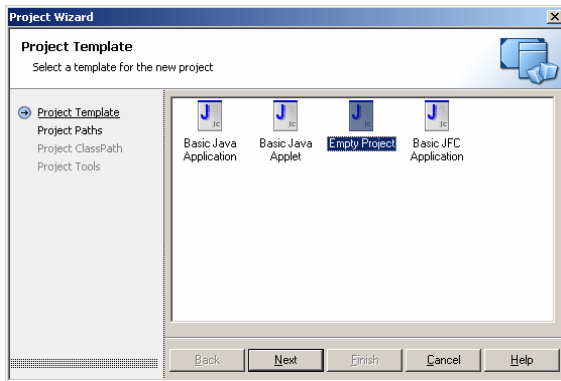
2. Phidgets and JCreator

2.1 Creating a simple program without GUI and without events

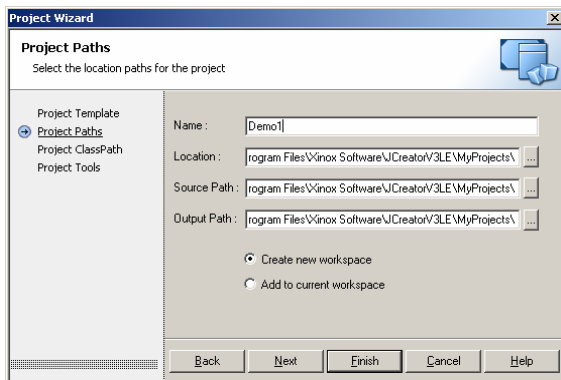
Start JCreator and create a new project (file > new project)



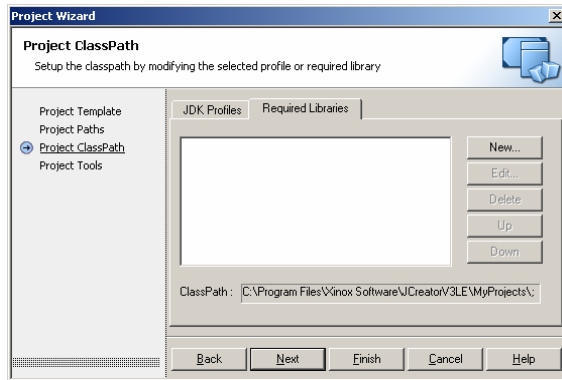
In the wizard that pops up, choose to create an empty project and click next.



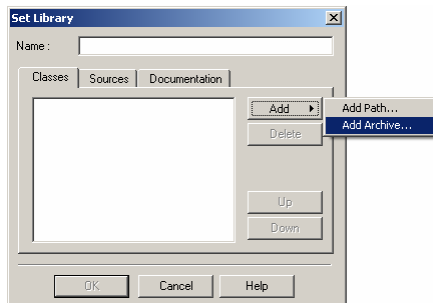
Provide a name for the project (for example: Demo1). Make sure that a new workplace is created for the project. Click next.



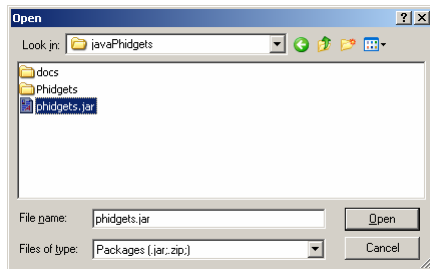
Add the phidget libraries to the project:
Go to the "Required Libraries" tab-page and click on "New".



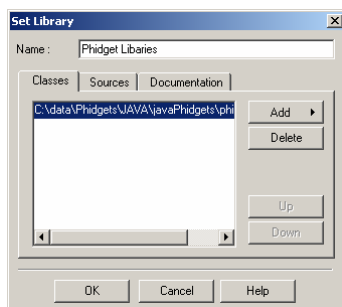
Select "Add Archive"



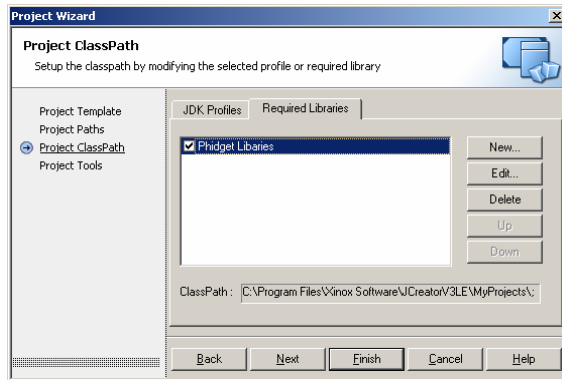
Browse to the directory where you installed the Phidget JAVA Libraries earlier (My Documents\Phidgets\Java\javaPhidgets) and select the file "Phidgets.jar"



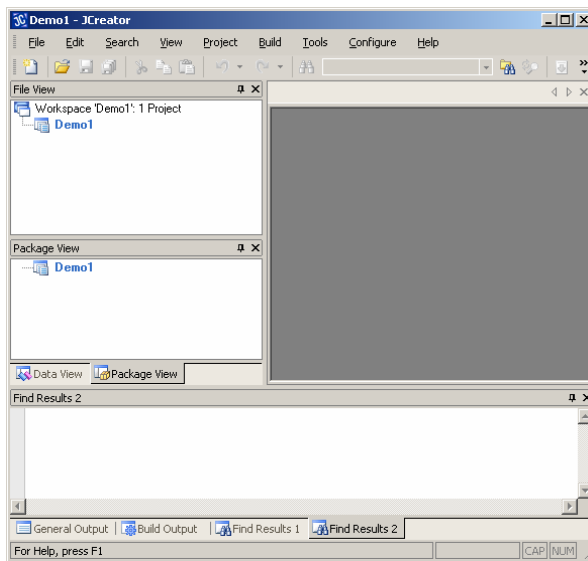
Provide a name for this set ("phidget libraries") and click ok



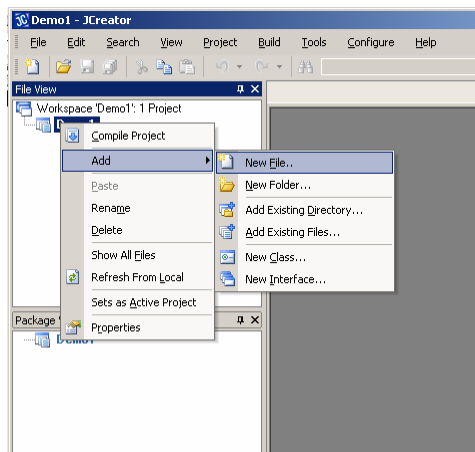
Make sure that you check this new set in the wizard before clicking on "Finish"



You created an empty project



Right click "Demo1" in the file view to add a new file and name it Demo1



Enter the following code in the window (read comments to understand program)

```
// Import phidget library
import Phidgets.*;

public class Demol
{
    //-----
    // Main
    //-----

    public static void main(String[] args) {

        // Create new instance of phidget interfacekit
        PhidgetInterfaceKit phid = new PhidgetInterfaceKit();

        // Connect to phidget webserver:
        // local computer, port 5001, password "pass"
        // any interfacekit / serialnumber irrelevant (-1)
        phid.OpenRemoteIP("localhost", 5001, -1, "pass");

        // Print result of 10 measurement to the command prompt

        int value;
        System.out.println("Let's do 10 measurements on port 0...\n");
        for(int i = 0; i < 10; ++i) {
            // Store measurement from pin 0 in variable "value"
            value = phid.GetSensorValue(0);
            //Print result
            System.out.println("Measurement " + i + ": " + value);
            // 250 msec pause
            Pause(250);
        }

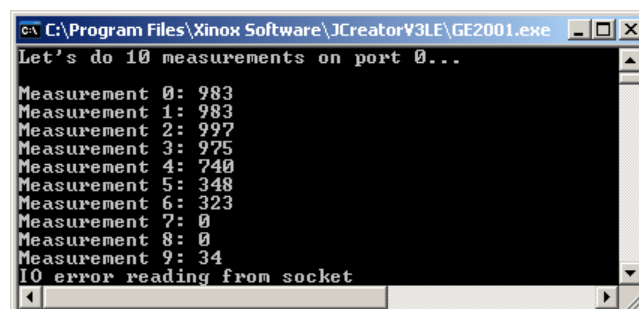
        // Close program (WARNIGN: the phid.Close() function might gene-
        // rate a socket closed warning in the command prompt. This is
        // probably a bug in the current JAVA implementation)

        phid.Close();
        System.out.println("Closed and exiting...");
    }

    //-----
    // Pause
    //-----

    public static void Pause(int msec)
    {
        // This function can be used to create a pause
        try {
            Thread.sleep(msec);
        } catch (InterruptedException e) {
            ;
        }
    }
}
}
```

Select from the menu build > Compile project (or press F7).
Press on the execute icon to run the program (or press F5).



As a next step, the example is extended with some physical outputs. A number of leds is switched on depending on the measurement value: led 0 is switched on if value is > 12.5% of the maximum, led 1 is switched on if value is > 25% of maximum value, etc, etc.

Replace the main function of the previous example with this code and study the changes in the code.

```
public static void main(String[] args) {

    // Create new instance of phidget interfacekit
    PhidgetInterfaceKit phid = new PhidgetInterfaceKit();

    // Connect to phidget webserver:
    // local computer, port 5001, password "pass"
    // any interfacekit / serialnumber irrelevant (-1)
    phid.OpenRemoteIP("localhost", 5001, -1, "pass");

    // Print result of 100 measurement to the command prompt
    // and show a led-bar effect

    int value;
    System.out.println("Do 100 measurements and create a led-bar\n");
    for(int i = 0; i < 100; ++i) {
        // Store measurement from pin 0 in variable "value"
        value = phid.GetSensorValue(0);
        // 100 msec pause
        Pause(100);
        //Print result
        System.out.println("Measurement " + i + ": " + value);
        // Show results in LED
        for (int j=0;j<8;j++) {
            if (value> (j*1000/8)) {
                phid.SetOutputState(j,true);
            } else {
                phid.SetOutputState(j,false);
            }
        }
    }

    // Close program (WARNIGN: the phid.Close() function might gene-
    // rate a socket closed warning in the command prompt. This is
    // probably a bug in the current JAVA implementation)

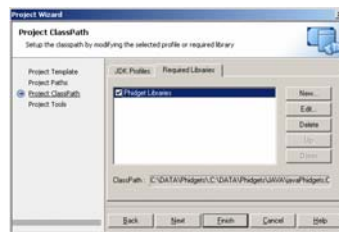
    phid.Close();
    System.out.println("Closed and exiting...");
}
```

2.1 Creating a simple program without GUI but with events

In the first examples, we read out the sensor values by ourselves at a predefined rate. This is called polling. In many cases, it can be more convenient to let the system decide when to take action based on special events.

Such an event-based approach will be demonstrated in the next example: the phidget driver monitors all values and gives a signal once a value changed too much. This means that the program doesn't need to monitor all values, but just only needs to respond to special (pre-configured) events once required.

Create a new empty project as was done in the previous example and name it Demo2. Note: instead of defining the "Phidget Library" path, you can simple check the required libraries since you already defined them in example 1.



Copy the following code to the project

```
import Phidgets.*;
import Phidgets.Events.*;

public class Demo2 implements _IPhidgetInterfaceKitEvents
{
    //-----
    // Special events
    //-----

    public void OnAttach(_IPhidgetEvents_OnAttachEvent ke) {
    }

    public void OnDetach(_IPhidgetEvents_OnDetachEvent ke) {
    }

    public void OnError(_IPhidgetEvents_OnErrorEvent ke) {
    }

    public void OnSensorChange(_IPhidgetInterfaceKitEvents_OnSensorChangeEvent ke) {
    }

    public void OnInputChange(_IPhidgetInterfaceKitEvents_OnInputChangeEvent e) {
    }

    //-----
    // Main
    //-----

    public static void main(String[] args) {
        new Demo2();
    }

    //-----
    // Demo2
    //-----

    public void Demo2() {
        System.out.println("Demo 2: Closed and exiting...");
    }
}
```

Looking at the code, you see that not only the phidget components are imported, but also information about special events that can be generated.

All events are defined in a so-called "JAVA Interface Object". In order to use these events in your code, you need to implement a copy of this interface element, by adding the statement "implements <objectname>" to the class definition.

In this example all possible events – like connecting a new device, sensor changer, etc - will be demonstrated, therefore the project will implement "_IPhidgetInterfaceKitEvents". In your own project you might only be interested in implementing events that detect sensor changes. For more information see the Phidget JAVA documentation.

Since we are implementing a "JAVA Interface Object" and not a derived "JAVA Class", the definition of all supported events need to be copied in your class. For an overview of the implemented classes you should refer to the Phidget JavaDocs, or just try to compile the program and reconstruct the methods from the generated error-messages.

Phidgets.Events

Interface _IPhidgetInterfaceKitEvents

All Superinterfaces:

[_IPhidgetEvents](#), [java.util.EventListener](#)

All Known Implementing Classes:

[_IPhidgetInterfaceKitEventsAdapter](#)

Method Summary

void	OnInputChange (_IPhidgetInterfaceKitEvents_OnInputChangeEvent e)
void	OnSensorChange (_IPhidgetInterfaceKitEvents_OnSensorChangeEvent e)

Methods inherited from interface [Phidgets.Events](#): [_IPhidgetEvents](#)

[OnAttach](#), [OnDetach](#), [OnError](#)

```
//-----  
// Special events  
//-----  
  
public void OnAttach(_IPhidgetEvents_OnAttachEvent ke) {  
}  
  
public void OnDetach(_IPhidgetEvents_OnDetachEvent ke) {  
}  
  
public void OnError(_IPhidgetEvents_OnErrorEvent ke) {  
}  
  
public void OnSensorChange(_IPhidgetInterfaceKitEvents_OnSensorChangeEvent  
ke) {  
}  
  
public void OnInputChange(_IPhidgetInterfaceKitEvents_OnInputChangeEvent e) {  
}
```

Instead of running all code from the main-function, it is good practice to put all code in a special function that is called from the main function. The how&why of this goes beyond of the scope of this tutorial, but you one can imagine that in this way extending and combining programs can be made easier.

```
//-----  
// Main  
//-----  
  
public static void main(String[] args) {  
    new Demo2();  
    System.out.println("New thread started. Press CTRL+C to quit...");  
}  
  
//-----  
// Demo2  
//-----  
  
public void Demo2() {  
    System.out.println("Demo 2: Closed and exiting...");  
}
```

Now let's start implementing the code.

The events that are generated because new phidget devices were plugged in, detached or other errors are occurred are not interesting since in a generic prototype this is not allowed to happen. Therefore, just put some error message in these event handlers.

```
public void OnAttach(_IPhidgetEvents_OnAttachEvent ke) {  
    System.out.println("New device attached...");  
}  
  
public void OnDetach(_IPhidgetEvents_OnDetachEvent ke) {  
    System.out.println("Device detached...");  
}  
  
public void OnError(_IPhidgetEvents_OnErrorEvent ke) {  
    System.out.println("An error occurred...");  
}
```

For the event caused by a change in the analog input, obtain the number of the input that changed and its new value and print these to the command line. Do a similar action for a new digital input.

```
public void OnSensorChange(_IPhidgetInterfaceKitEvents_OnSensorChangeEvent ke)  
{  
    int index = ke.get_Index();  
    int value = ke.get_SensorValue();  
    System.out.println("Analog input " + index + "changed: " + value);  
}  
  
public void OnInputChange(_IPhidgetInterfaceKitEvents_OnInputChangeEvent e) {  
    int index = e.get_Index();  
    boolean value = e.get_NewState();  
    System.out.println("Digital input " + index + "changed: " + value);  
}
```

In the startup code of the class, the phidget component is not only initialized as in the previous example, but also the events are configured. For example, you can configure that an OnSensorChange event is only generated after a change on the input of at least 10.

```
public Demo2() {
    // Create new instance of phidget interfacekit
    PhidgetInterfaceKit phid = new PhidgetInterfaceKit();

    // Connect to phidget webserver:
    // local computer, port 5001, password "pass"
    // any interfacekit / serialnumber irrelevant (-1)
    phid.OpenRemoteIP("localhost", 5001, -1, "pass");

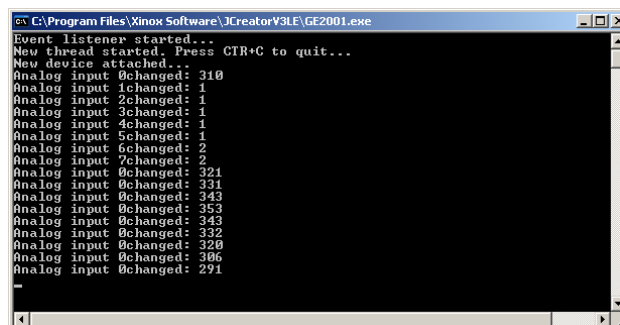
    // Set sensitivity of analog input 0
    // Only create OnSensorChangeEvents after sensor changed at least 10
    phid.SetSensorChangeTrigger(0,10);

    // Add event listener
    phid.add_IPhidgetInterfaceKitEventsListener(this);
    System.out.println("Event listener started...");
}
```

The full code is listed below:

After starting the program the following events happen:

- The main functions creates a new copy of the Demo2()-class that runs as a separate thread / separate program. Although the main functions has finished, the newly started thread is still running as an independent program....
- The Demo2() function, initializes the phidgetInterfaceKit and its associated events. After finishing the initialization, events are automatically generated when inputs change.
- Events are handled once input change, independently of the main program. This means that although the main program might appear to have finished, the thread that handles the events is still operational. You can only quit the program by pressing CTRL+C.



```
C:\Program Files\Xinow Software\JCreatorV3LE\GE2001.exe
Event listener started...
New thread started. Press CTRL+C to quit...
New device attached...
Analog input 0changed: 310
Analog input 1changed: 1
Analog input 2changed: 1
Analog input 3changed: 1
Analog input 4changed: 1
Analog input 5changed: 1
Analog input 6changed: 2
Analog input 7changed: 2
Analog input 0changed: 321
Analog input 0changed: 331
Analog input 0changed: 343
Analog input 0changed: 353
Analog input 0changed: 343
Analog input 0changed: 332
Analog input 0changed: 320
Analog input 0changed: 306
Analog input 0changed: 291
```

```

// Import phidget library
import Phidgets.*;
import Phidgets.Events.*;

public class Demo2 implements _IPhidgetInterfaceKitEvents
{
    //-----
    // Special events
    //-----

    public void OnAttach(_IPhidgetEvents_OnAttachEvent ke) {
        System.out.println("New device attached...");
    }

    public void OnDetach(_IPhidgetEvents_OnDetachEvent ke) {
        System.out.println("Device detached...");
    }

    public void OnError(_IPhidgetEvents_OnErrorEvent ke) {
        System.out.println("An error occured...");
    }

    public void OnSensorChange(_IPhidgetInterfaceKitEvents_OnSensorChangeEvent
ke) {
        int index = ke.get_Index();
        int value = ke.get_SensorValue();
        System.out.println("Analog input " + index + "changed: " + value);
    }

    public void OnInputChange(_IPhidgetInterfaceKitEvents_OnInputChangeEvent e) {
        int index = e.get_Index();
        boolean value = e.get_NewState();
        System.out.println("Digital input " + index + "changed: " + value);
    }

    //-----
    // Main
    //-----

    public static void main(String[] args) {
        new Demo2();
        System.out.println("New thread started. Press CTRL+C to quit...");
    }

    //-----
    // Demo2
    //-----

    public Demo2() {
        // Create new instance of phidget interfacekit
        PhidgetInterfaceKit phid = new PhidgetInterfaceKit();

        // Connect to phidget webserver:
        // local computer, port 5001, password "pass"
        // any interfacekit / serialnumber irrelevant (-1)
        phid.OpenRemoteIP("localhost", 5001, -1, "pass");

        // Set sensitivity of analog input 0
        // Only create OnSensorChangeEvents after sensor changed at least 10
        phid.SetSensorChangeTrigger(0,10);

        // Add event listener
        phid.add_IPhidgetInterfaceKitEventsListener(this);
        System.out.println("Event listener started...");
    }
}

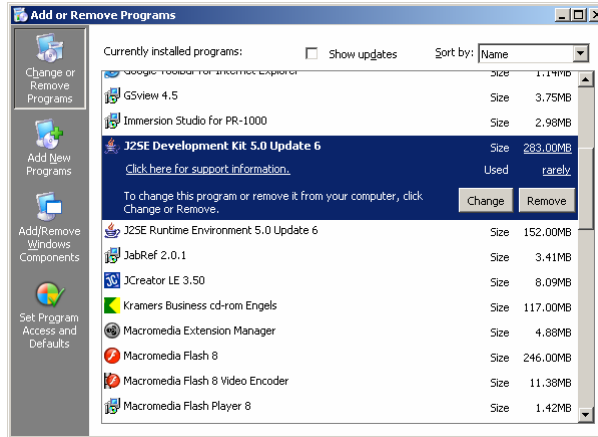
```

3. Common Errors / FAQ

3.1 Help! JCreator can't find the JAVA compiler

Step 1: Make sure that a JAVA Software Developer Kit is installed (NOT only Runtime!!!).

Go to windows Start > Control Panel > Add or Remove Programs and check if there is a Developer kit installed. If not, download J2SE from www.sun.com.

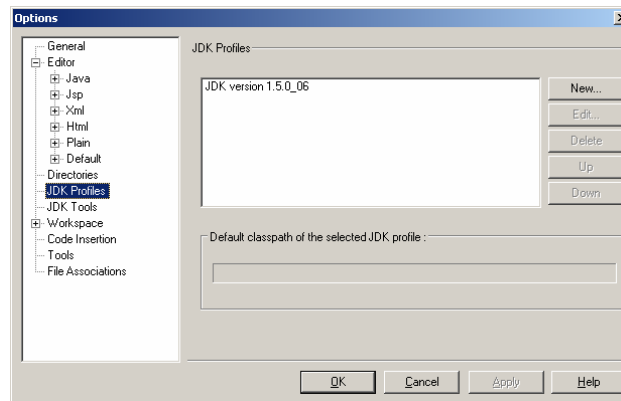


Step 2: Make sure that the JAVA Software Developer Kit is recognized by JCreator.

Start JCreator and go to configure > options

Make sure that a JDK profile has been created for your Java SDK.

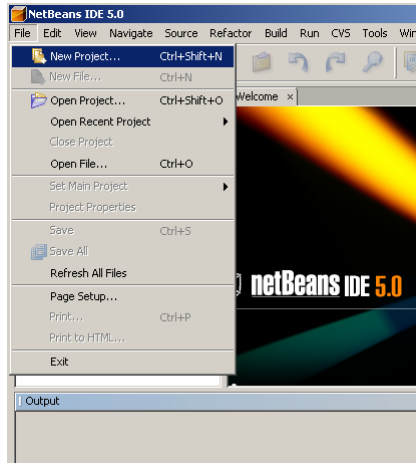
If no profile is available, uninstall Jcreator and re-install. During the install procedure, make JCreator create an automatic profile (otherwise you need to create one manually after installation).



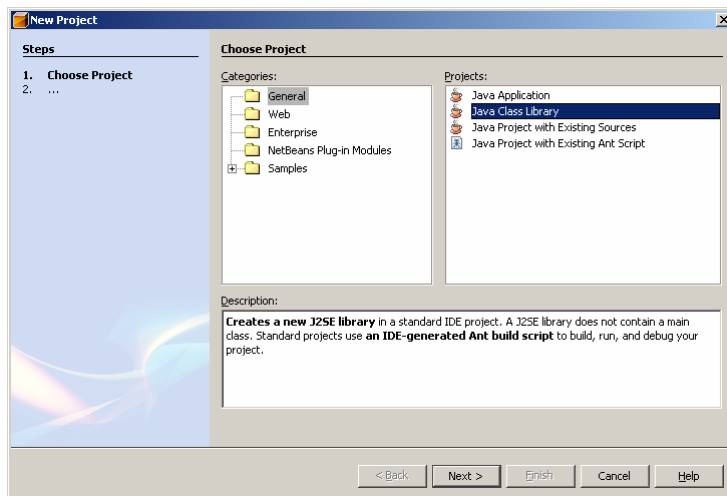
A. Phidgets and Netbeans

This example only explains how to setup a phidget project in netbeans. For details about the code, you are referred to the fully worked out example in JCreator.

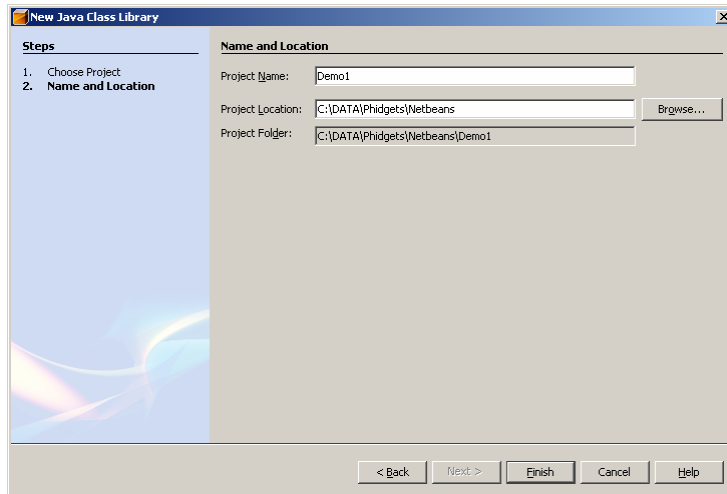
Create a new project (menu start > new project)



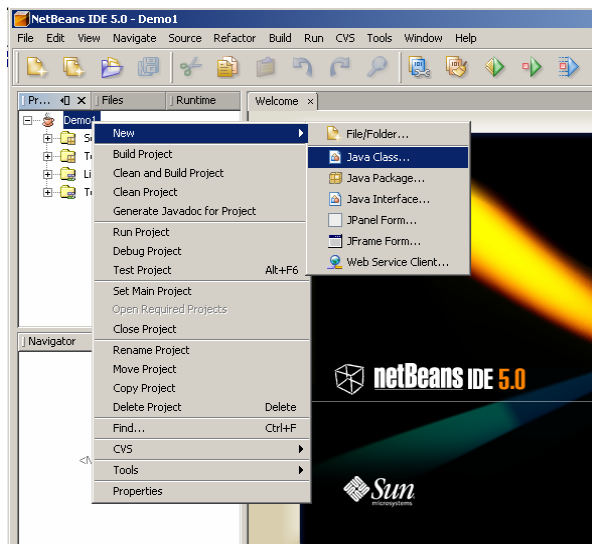
In the wizard, select to create a Java Class Library, this means, an application without a graphic user interface. Click Next.



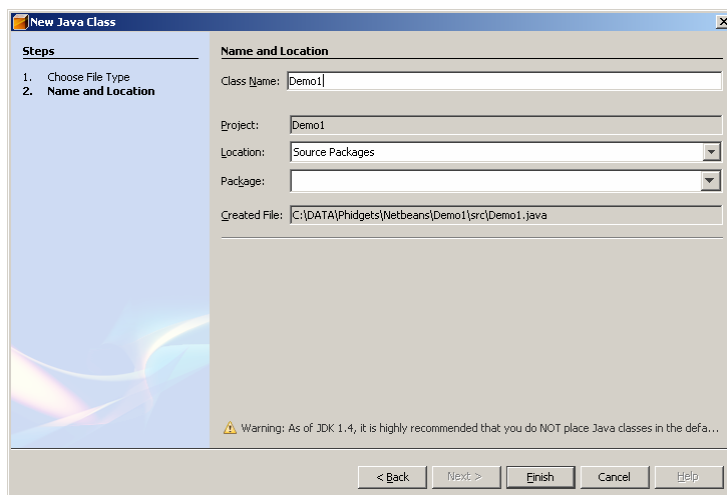
Select an appropriate location for the project (browse) and create a project called Demo1. Click Finish.



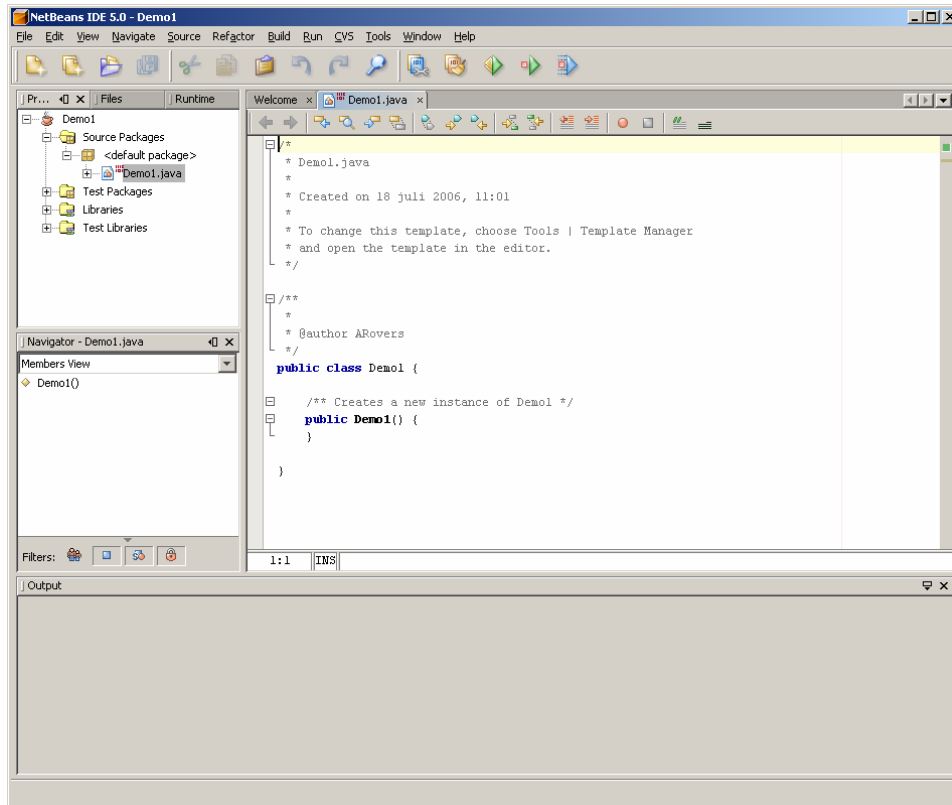
An empty project structure is shown now.
Select the Demo1 main folder and add a new Java Class by clicking with the right mouse button.



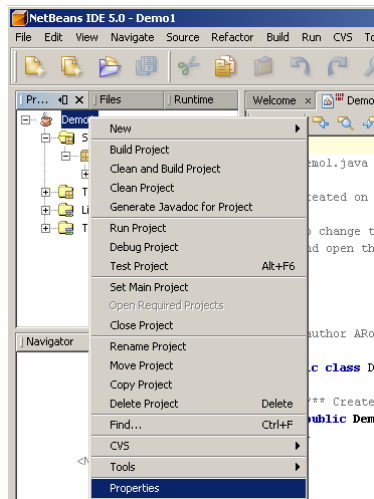
Name the file Demo1 and click finish.



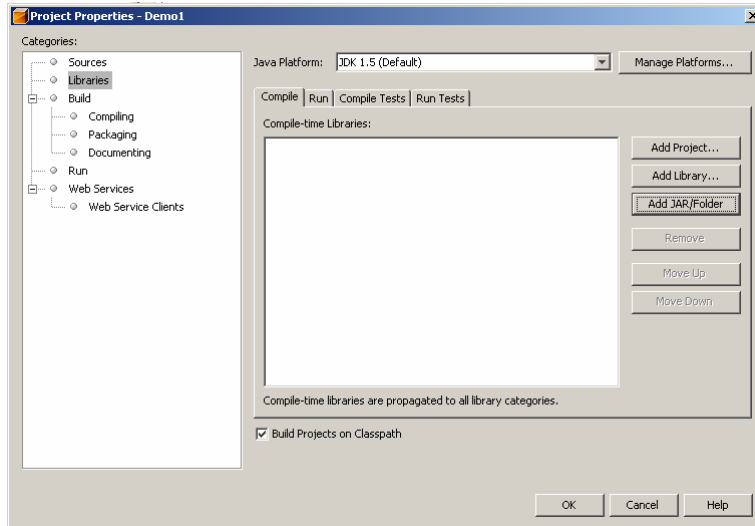
An empty template is already created for you.



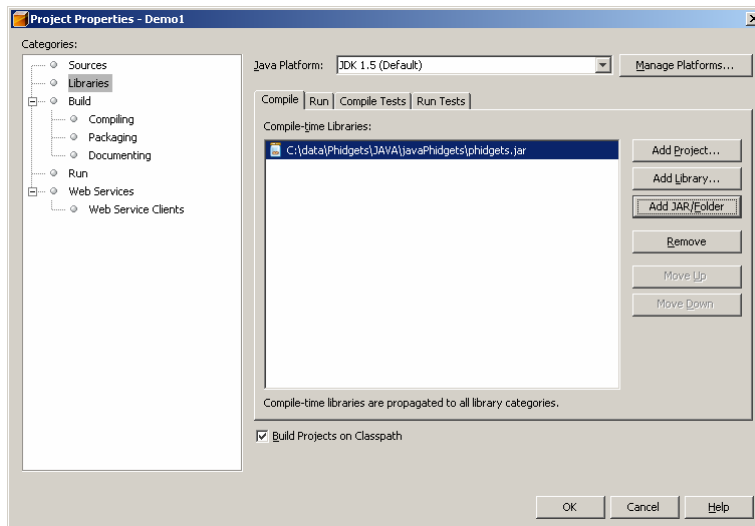
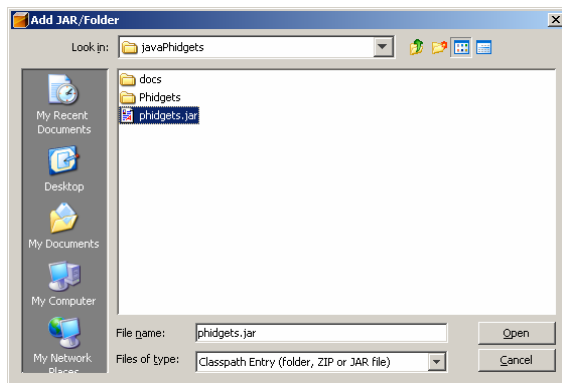
Now add the phidget libraries to the project.
Again, select the Demo1 main folder in the project view and by using the right mouse button, select "Properties".



Go to the "Libraries" tab and select "Add JAR/Folder" under compile options.

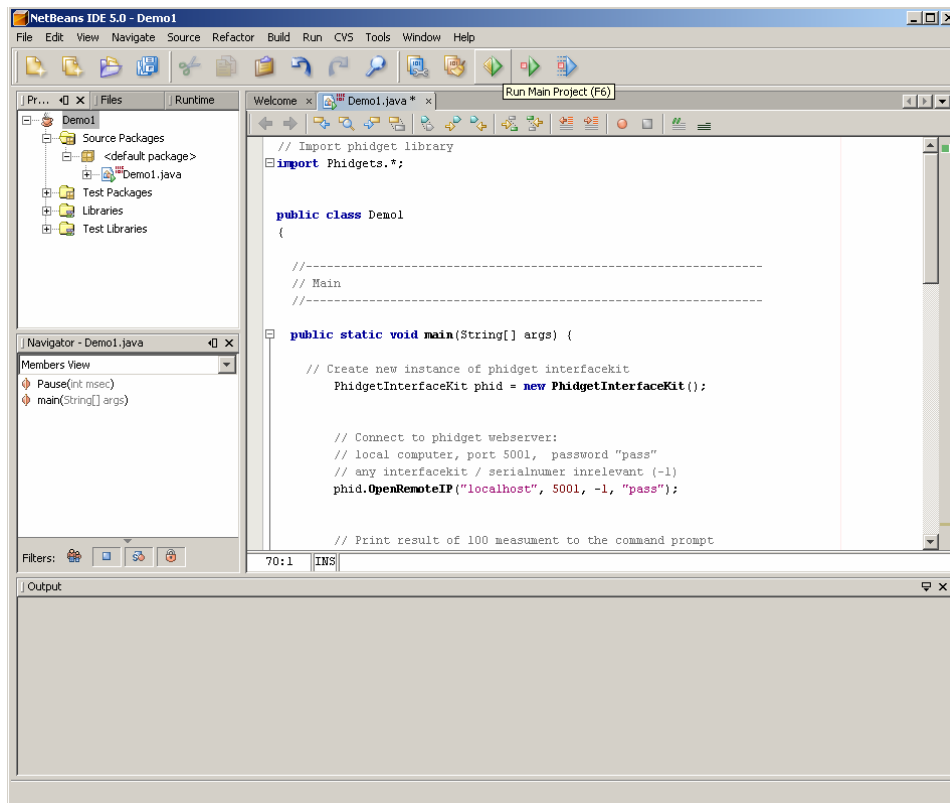


Browse to the directory where you installed the Phidget JAVA Libraries earlier (My Documents\Phidgets\Java\javaPhidgets) and select the file "Phidgets.jar"

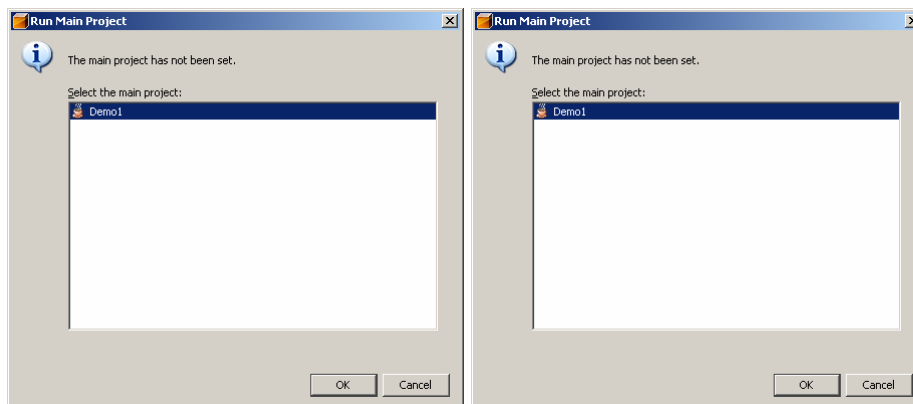


Click OK to return to the project.

Your project is ready to be used with phidgets. Copy the code from the JCreator example to the Demo1 file and press the Run button (or F6) to compile and run the project.



The first time the project is run, Netbeans asks for the main project and main class. Select Demo1 and press OK (twice).



Note: In netbeans the application is not run in a windows command prompt. Instead all output is reredericted to the netbeans prompt below your code.

