# JDroidLib & JTurtleLib Installation & Use

***Version 2.00, for the Windows OS, differences to Linux OS and Mac OS are annotated***

The *JDroidLib* and the JTurtleLib frameworks are distributed as ZIP archive and contain the following subfolders:

- **App**: Sample apps for Android smartphones/tablets that supports Android V2.2 up
- **Tutorial**: Apps that are part of the JDroidLib or JTurtleLib tutorial on website www.aplu.ch/android or www.aplu.ch/jturtlelib
- **Lib**: JDroidLib.jar/TcpALib.jar or JTurtleLib.jar library files
- **Doc**: The JavaDoc for JDroidLib/TcpALib or JTurtleLib
- **ProjectBuilder**: the GUI version of the project builder, including source
- **LogCat**: a user friendly interface to the Android Debug Logger
- **InstallUSBLinux**: USB device installer for Android smartphones (under Linux)
- **InstallationAndUse:** This installation guide

**Preliminary**: In any case the Java Runtime Environment (**JRE**) must be installed

**Important**: If you use our Online-Editor/Compiler (at *http://www.java-online.ch*) and you download the app directly to the smartphone, **no additional installations are necessary**. Try it out and enjoy the feeling of success to create your first Android app even if you are a Java beginner.

(Moreover from *http://clab2.phbern.ch/jOnline/webstart/InstallEmul.jnlp* a slim-emulator version is installed in the subfolder *.jdroidemul* of the user home directory. There you will find the starter application *ExecEmul.jar*.)

# Eclipse IDE

(If you need help, consult the many installation guides on the Internet)

**Installation**

- Download and install the most recent version JDK SE in any folder (called <*idehome*> here), see *http://www.oracle.com/technetwork/java/javase/downloads*

- (Windows only:) Set an environment variable *JAVA_HOME* that points to <*idehome*> and add <*idehome*>*bin* to the path

- Test the installation by opening a command shell and type *javac*. A usage information must be shown

- Download the Android SDK from *http://developer.android.com/sdk*. Unpack the zip/tgz in any folder, called <*androidsdk*> here. (Windows: Add <*sdkhome*>*tools* and <*sdkhome*>*platform-tools to the path)*

- Start the Android-SDK Manager (Windows: *SDK Manager.exe* in <sdkhome>, Linux, Mac: *android* in <sdkhome>/tools, start with command *./android*). Choose *package to install*. For a minimum installation remove all options but *SDK-Android SDK-Tools Version 6* and *Platform 2.2.* Wait for the download and installation of these options

- With the Android-SDK Manager select *Virtual devices* and *New*. Make the following entries*:*

Name: MyEmulator
Target: Android 2.2
SD Card: 2000 MiB
Build-In: HVGA

Click *Create* and wait for the confirmation message. Start the emulator by clicking the Start button (you need some patience).

- Download Eclipse from *http://www.eclipse.org/downloads* (e.g. version *Eclipse IDE for Java Developers*). Unpack it it in any folder. Start Eclipse.

- Install the ADT plugin:

  *Help* | *Install Software*
  Click button *Add*
  Enter any identifier name
  Enter the location `https://dl-ssl.google.com/android/eclipse`
  Click Ok and select *Developer Tools*
  Click *Next* to start the installation of the plugin

- Restart Eclipse and select *Window* | *Preferences*. Select *Android* and enter the path to the Android-SDK (<sdkhome>)

- Download the *JDroidLib* or *JTurtleLib* distribution from *http://www.aplu/android* or http://www.aplu.ch/jturtlelib and unpack it in any folder. Copy *JDroidLib.jar* or *JTurtleLib* in some folder where you store your library jars. Copy *ProjectBuilder.jar* in any folder where you store special programs/applications and create a link to start the ProjectBuilder from the desktop or application launch bar (Linux/Mac: Give ProjectBuilder.jar executable permission by typing *chmod +x ProjectBuilder.jar*. Right click and select *Open with Sun Java Runtime*)

**Use:**

- *File* | *New* | *Other*
  *Choose Project:* **Android | Android Project**
  *Project name*: **Apps** (example)
  *Build target:* **Android 2.2**
  *Application name:* **MyApp1** (example)
  *Package name:* **ch.aplu.app** (example)
  *Create Activity*: **remove selection**
  Finish

- Start the ProjectBuilder (create a link to ProjectBuilder.jar). Enter in text fields:
  **Project Root**: your project root folder
  **Package Name**: ch.aplu.app (example)
  **App Name**: MyApp1 (example)
  **Library File(s)**: the fully qualified path to JDroidLib.jar or JTurtleLib (not both!), other libraries separated by semicolon
  **Sprite Folder**: the fully qualified path to your sprite image files (if you want to change the App logo, copy your own jdroid_gglogo.png in this folder)
  **Media Folder**: the fully qualified path to your media files (e.g. wav, mp3)
  *Build*

- **Press F5 (Refresh).** You see *MyApp1.java* in the Package Explorer.

- In the Package Explorer right-click on the project. Select *Properties* | *Resources* | *Java Build Path* | *Libraries* | *Add External Jars* and select *JDroidLib.jar or JTurtleLib* found in <projectroot>\libs
  Press Ok

- Connect your smartphone or start the Android emulator. Press the *Run* button and select Android Application. The project will be compiled, packed, signed and installed and hopefully started.

.

# Netbeans IDE

(If you need help, consult the many installation guides on the Internet. We recommend to use Eclipse on Linux/Mac platforms. The following guide is for Windows only)

**Installation**

- Download and Install the most recent version JDK SE in any folder (called <*idehome*> here), see
  *http://www.oracle.com/technetwork/java/javase/downloads/index.html*
  Set an environment variable: *JAVA_HOME* that points to <*idehome*> and add <*idehome*>*bin* to the path

- Download the Android SDK from *http://developer.android.com/sdk/index.html*
  Unpack the zip in any folder, e.g. *c:\programs\android-sdk* (called <*androidsdk*> here).
  Add <*sdkhome*>*tools* and <*sdkhome*>*platform-tools* to the path

- Start the *Android-SDK Manager* (found in <*sdkhome*>). Choose *package to install*. For a minimum installation remove all options but *SDK-Android SDK-Tools Version 6* and *Platform 2.2.* Wait for the download and installation of these options.

- With the Android-SDK Manager select *Virtual devices* and *New*. Make the following entries*:*

Name: MyEmulator
Target: Android 2.2
SD Card: 2000 MiB
Build-In: HVGA

Click *Create* and wait for the confirmation message. Start the emulator by clicking the Start button (you need some patience).

- Download Netbeans from http://netbeans.org/downloads (version *Java SE is enough*) and install it in any folder

- Download Ant from *http://ant.apache.org* (binary distribution) and unpack it in any folder (called <*anthome*>). Set an environment variable *ANT_HOME* that points to <*anthome*> and add <*anthome*>*bin* to the path

- Install the Netbeans Android plugin:

  Start Netbeans
  Go to *Tools* | *Plugins* | *Settings* and click *Add*
  Enter any name and the URL:
  `http://kenai.com/downloads/nbandroid/updatecenter/updates.xml`
  Go to *Plugins* | *Available Plugins* and search for "android"
  Select *Android* and the *Test Runner* version that corresponds to the installed Netbeans version
  Press *Install* and confirm everything

- Download the *JDroidLib* or *JTurtleLib* distribution from *http://www.aplu.ch/android* or http://www.aplu.ch/jturtlelib and unpack it in any folder. Copy *JDroidLib.jar* or *JTurtleLib* in some folder where you store your library jars. Copy *ProjectBuilder.jar* in any folder where you store special programs/applications and create a link to start the ProjectBuilder from the desktop or application launch bar.

**Use:**

- *File* | *New Project*
  *Choose Project:* **Android | Android Project**
  Press *Next*

- Enter *Project name*: **Apps** (example)
  Select *Set as Main Project*
  *Package name:* **ch.aplu.app** (example)
  *Activity Name*: **MainActivity**
  If you never did it before, click button *Manage Android SDK* and select the Android SDK home directory (<sdkhome>)
  *Target Platform*: **Android 2.2**
  *Finish*

- Start the ProjectBuilder. Enter in text fields:
  **Project Root**: your project root folder
  **Package Name**: ch.aplu.app (example)
  **App Name**: MyApp1 (example)
  **Library File(s)**: the fully qualified path to JDroidLib.jar or JTurtleLib.jar (not both), other libraries separated by semicolon
  **Sprite Folder**: the fully qualified path to your sprite image files files (if you want to change the App logo, copy your own jdroid_gglogo.png in this folder)
  **Media Folder**: the fully qualified path to your media files (e.g. wav, mp3)
  *Build*

- Right-click the project in the Netbeans *Project Manager* and select *Clean and Build*. The Android app is created and the error messages should disappear. Under Source Packages you find the template source *MyApp1.java* and *MainActivity.java*. *MainActivity.java* is never used and can be deleted. If the Project is selected as *Main Project*, you may click the title bar *Run* button to compile/pack/sign/download and start the app (in some circumstances you need to click twice the Run button).

.

# BlueJ (and other Java Editors/IDEs)

(If you need help, consult the many installation guides on the Internet. We recommend to use Eclipse on Linux/Mac platforms. The following guide is for Windows only)

**Installation**

- Download and Install the most recent version JDK SE in any folder (called <*idehome*> here), see *http://www.oracle.com/technetwork/java/javase/downloads/index.html*
  Set an environment variable *JAVA_HOME* that points to <*idehome*> and add <*idehome*>*bin* to the path

- Download the Android SDK from *http://developer.android.com/sdk/index.html*
  Unpack the zip in any folder, e.g. *c:\programs\android-sdk* (called <*androidsdk*> here).
  Add <*sdkhome*>*tools* and <*sdkhome*>*platform-tools* to the path

- Start the *Android-SDK Manager* (found in <*sdkhome*>). Choose *package to install*. For a minimum installation remove all options but *SDK-Android SDK-Tools Version 6* and *Platform 2.2.* Wait for the download and installation of these options.

- With the Android-SDK Manager select *Virtual devices* and *New*. Make the following entries*:*

Name: MyEmulator
Target: Android 2.2

SD Card: 2000 MiB
Build-In: HVGA

Click *Create* and wait for the confirmation message. Start the emulator by clicking the Start button (you need some patience).

- Download Ant from *http://ant.apache.org* (binary distribution) and unpack it in any folder (called <*anthome*>). Set an environment variable *ANT_HOME* that points to <*anthome*> and add <*anthome*>*bin* to the path

- Download the *JDroidLib* or *JTurtleLib* distribution from *http://www.aplu.ch/android* or *http://www.aplu.ch/jturtlelib* and unpack it in any folder. Copy *JDroidLib.jar* or *JTurtleLib.jar* in some folder where you store your library jars. Copy *ProjectBuilder.jar* in any folder where you store special programs/applications and create a link to start the ProjectBuilder from the desktop or application launch bar.

- Download BlueJ from *http://www.bluej.org* and install it in any folder. Start BlueJ and select *Tools* | *Libraries*. Click *Add* and insert the fully qualified path to *JDroidLib.jar* or *JTurtleLib* and to *android.jar* in the platform\android-8 subfolder of the installed Android-SDK, e.g. <sdkhome>\platforms\android-8\android.jar.

- Create a folder where you put your BlueJ projects, e.g. e:\mybj.

**Use:**

- With the file explorer create a new project folder in your BlueJ project folder, e.g. *MyApp*

- Start BlueJ. Select *Project* | *New Project* and choose the folder *e:\mybj\MyApp (example)*. Enter *src* as filename and press *Create*
.
- Select *Edit* | *New Package* and enter a package name, e.g. *ch.aplu.ex*. Press *Ok*.

- In a command shell execute the following command (adapt to your example, all on one line):

  ```
  android create project --target 1 --path e:\mybj\MyApp
  --activity MyApp1 --package ch.aplu.ex
  ```

  (the target number corresponds to the id of the Android version that depends how you installed the Android SDK. Perform command

  ```
  android list targets
  ```

  to find out which id corresponds to Android 2.2)

- Start the ProjectBuilder. Enter in text fields:
  **Project Root**: e:\mybj\MyApp (example)
  **Package Name**: ch.aplu.ex (example)

**App Name**: MyApp1 (example)
**Library File(s)**: the fully qualified path to JDroidLib.jar or JTurtleLib.jar (not both!), other libraries separated by semicolon
**Sprite Folder**: the fully qualified path to your sprite image files (if you want to change the App logo, copy your own jdroid_gglogo.png in this folder)
**Media Folder**: the fully qualified path to your media files (e.g. wav, mp3)
*Build*
Click *Replace* in the warning dialog

- *MyApp1.*java is now modified to the the JDroidLib or JTurtleLib template. In BlueJ select the folder *ch*, then *aplu*, then *ex*. Select *Menu Edit* | *Add class from file* and search for *MyApp1*. The *MyApp1* icon appears in the project window  It may be double-clicked to show it in the BlueJ editor and should compile without errors

- In a command shell select the project root as current directory (e.g. e:\mynb\MyApp). Type

  ```
  ant debug
  ```

  and ant will compile/pack/sign your Android app. MyApp1-debug.apk is found in the subfolder bin.

- Select the bin subfolder as current directory and type

  ```
  adb install -r MyApp1-debug.apk
  ```

  to install it on a running emulator or USB-connected smartphone

- Start the app on the emulator or the smartphone

- Modify the source in BlueJ, press *Compile* and repeat the steps with *ant debug* and *adb install*. The new app should start

- You may add more classes to the project by selecting *Menu Edit* | *New Class* in the ch.aplu.ex1 project view, where *MyApp1* resides. If you need sprite images, select the *Sprites Folder* in the ProjectBuilder, where the sprites resides and click *Build* and then *Keep*

**Remarks**:

- The command line usage could be avoided by creating batch files

- As you see in this installation, only the editor of BlueJ is actually used, the IDE compilation only serves to show syntax errors because `ant debug` recompiles the source. Therefore any other Java editor (**JCreater, Java Editor**, etc.) may be configured to develop JDroidLib apps analogously. (The simple scripting facilities of JCreator may automate the building process favorably.)

## Hints to install apps via USB:

- **Windows:** Install the brand specific synchronization software, e.g. for HTC: *HTC Synch*, for Samsung *Kies*. This will also install the USB drivers as well (otherwise hard to find and install)
  **Linux:** Execute *InstallUSBLinux.jar* found in the JDroidLib or JTurtleLib distribution will install drivers for many Android devices of currently known manufacturers.
  **Mac:** Nothing to install, devices should be found automatically

- Setup your smartphone:
  - Applications | Unknown sources: selected
  - Development | USB debugging: selected
  - Development | Stay awake: selected
  - **Do not** enable your smartphone as external disk or as GSM modem (for HTC, select Settings | PC connection | Charging only)
  - Shutdown and restart the smartphone

- To test the USB connection, start a command shell and type

  *adb devices* (Linux/Mac: *./adb devices*)

  The smartphone ID must be displayed. If no device is listed, shutdown the PC and the smartphone and unplug the USB connection. Restart the PC and the smartphone and reconnect the smartphone to the USB port.

- Manual installation of Android apps: Start a command shell and go to the folder where the application apk resides. Type

  *adb install -r <appname.apk>*
  (Linux/Mac: *./adb -r <appname.apk>*)

## How to use the Debug Console

LogCat is GUI based shell to the Android Debugging facility. It assumes that adb (and for Windows some additional DLLs) resides in the subdirectory *.jdroidtools* of user home directory <*userhome*>*.*

When startet, it first checks if there is an attached device (emulator or real device). If no device is found, the program exits with an error message.

If a device is found, the command *adb logcat* <*filter*> is spawned, where <*filter*> is the only command line option. If no command line option is given, *adb logcat* is spawned.

The starter applications *ExecDebugNNN.jar* spawns LogCat.jar with the following filters:

*ExecDebugAplu.jar*: *:S ch.aplu.android:V* (displays JDroidLib/JTurtleLib message only)
*ExecDebugRT.jar*: *:S AndroidRuntime:E ch.aplu.android:V* (also displays realtime messages)
*ExecDebugAll.jar* : no filter (displays all messages)

**Installation:**
Copy the distributed folder *jdroidtoolsXXX* (XXX indicates your OS) as *.jdroidtools* (don't forget the leading period) in your home directory and create links to the *ExecDebugNNN.jar* (or call *LogCat.jar* with the filter you want).

## Supplements:

- To change the app icon, replace *icon.png* in the following folders:

  <projectroot>\res\drawable-hdpi (72x72 dpi)
  <projectroot>\res\drawable-mdpi (48x48 dpi)
  <projectroot>\res\drawable-ldpi (36x36 dpi)

- If your app is not installed or does not start, check whether another app with the same name or the same package is already installed. (This may cause a signature error if the app were developed and downloaded from a different computer). Remove the app and reinstall it.

- You may create several apps within the same project. Select the current app name in the ProjectBuilder and press *Build.*

- To add sprite images to the application, copy the files in the sprite folder as selected in the ProjectBuilder and press *Build.*

- To avoid the compiler warning on some systems ***warning 'includeantruntime' was not set***, add a new line in <sdk-root>\tools\ant\main_rules.xml in the section <javac encoding= *includeantruntime="false"*

- In details the **ProjectBuilder** performs the following tasks:
  - Creates a AndroidManifest.xml adapted to JDroidLib/JTurtleLib apps
  - Replaces <projectroot>\res\layout\main.xml
  - Creates some layout resources in <projectroot>\res\layout
  - Replaces <projectroot>\res\values\strings.xml
  - Creates a template source file <projectroot>\src\<packagepath>\<ApplicationName>.java
  - Copies the library files in <projectroot>\libs
  - Copies all files from <Sprites Folder> to <projectroot>\res\drawable
  - Copies all files from <Media Folder> to <projectroot>\res\raw
  - Copies jdroid_gglogo.png/turtle_white.gif to <projectroot>\res\drawable
  - Replaces <ApplicationName> in build.xml

- To change the turtle image, put your own turtle_white.gif in the sprites folder. White pixels will take the current turtle color (background: transparent pixels)