The JNLua Java VM module is a Lua module written in C that allows a Lua process to create a *Java Virtual Machine* and run Java code in that machine. In essence, the Java VM module supports bootstrapping JNLua from the Lua side.

Prerequisites

In order to use the Java VM module, the following prerequsites must be met:

- The JNLua Java VM module is called `javavm`. The module must be on the Lua C path, i.e. `package.cpath`. Pre-built versions for the Win32 platform (64-bit and 32-bit) are prorived in the `native` assembly. Please see Building the Native Library for information on how to build the Java VM module yourself.
- The Java virtual machine library and its dependent libraries must be on the system library path. The exact steps required to ensure this depend on the platform and the Java virtual machine, for example:
  - Win32: Add `{JDK}\jre\bin` and `{JDK}\jre\bin\server` to `PATH`.
  - Linux: Add `{JDK}/jre/lib/{ARCH}` and `{JDK}/jre/lib/{ARCH}/server` to `LD_LIBRARY_PATH`.
- The JNLua Java library, i.e. `jnlua-{version}.jar`, must be installed.

Example

The following Lua code provides an example of using the Java VM module:

```
javavm = require("javavm")
javavm.create("-Djava.class.path=jnlua-1.0.2.jar")
System = java.require("java.lang.System")
System.out:println("Hello, world!")
javavm.destroy()
```

Limitations

The JNI specification allows for only one Java virtual machine to be concurrently attached to a thread. Therefore, the Java VM module supports only one virtual machine at any point in time.

In practice, Java virtual machine implementations can be even more restrictive and allow only one Java virtual machine to be created for the lifetime of a process, i.e. even after the Java virtual machine has been destroyed, subsequent attempts of creating a new Java virtual machine fail.

Functions

The Java VM module provides the following functions.

javavm.create

Syntax:

```
vm = javavm.create({ option })
```

The function creates a Java virtual machine using the options provided and returns the virtual machine. In addition, the JNLua Java Module is loaded into the Lua state.

If there already is a virtual machine, the function raises an error.

Example:

```
javavm.create("-Djava.class.path=jnlua-1.0.1.jar")
```

javavm.get

Syntax:

```
vm = javavm.get()
```

The function returns the current virtual machine, or `nil` if there is none.

javavm.destroy

Syntax:

```
success = javavm.destroy()
```

The function destroys the current virtual machine and returns a boolean indicating whether the operation was successful. Calling the function when there is no current virtual machine returns `false`.