Srivus Scientifica Advanced Calculator System — Installation Guide

For students. This guide gets the Srivus Scientifica Advanced Calculator System running on your computer. It takes about ten minutes, and most of that is Java downloading.

You need two things: **Java**, and **the right jar file for your computer**.

Step 1 — Work out which file you need

You were given two files. You only need **one**. Which one depends on your computer:

Your computer	The file you need
Windows (any)	`ScientificCalculator-win-linux-intelmac.jar`
Linux	`ScientificCalculator-win-linux-intelmac.jar`
Mac with an **Intel** processor	`ScientificCalculator-win-linux-intelmac.jar`
Mac with **Apple silicon** (M1, M2, M3, M4)	`ScientificCalculator-mac-aarch64.jar`

Which Mac do I have?

Click the **apple menu** (top-left of the screen) → **About This Mac**.

- Says **Chip: Apple M1 / M2 / M3 / M4** → you have **Apple silicon** → use the `mac-aarch64` file
- Says **Processor: Intel ...** → you have an **Intel** Mac → use the `win-linux-intelmac` file

Every Mac sold since late 2020 is Apple silicon, so this is most likely you.

> **Apple silicon and Intel Macs need different files.** Picking the wrong one gives a confusing error about a library that cannot load. If the app will not start on a Mac, check this first.

> **Do your files have `-Basic-` in the name?** Then you have the free **Basic Edition**, and everything in this guide applies unchanged: the same platform choice, the same Java. Use the matching Basic file: `ScientificCalculator-Basic-win-linux-intelmac.jar` on Windows, Linux or an Intel Mac, and `ScientificCalculator-Basic-mac-aarch64.jar` on Apple silicon. The Basic Edition is a complete calculator for mathematics from the middle grades through Algebra 2 and Precalculus, and it has the whole language **and the Graphs tab** in it. The **Advanced Edition** adds nine tabs — Notes, 2D Plots, 3D Plots, Matrices, Stats, Distributions, WMW Test, Images and Chem — for AP Statistics, for chemistry and physics, for teachers, and for programming with the app. In Basic, those nine open as a page describing what the tab does. Both editions run the same language, so a file saved in one opens in the other.

Step 2 — Install Java

The calculator needs **Java 23 or newer**.

> **Java 17 and Java 21 will NOT work**, even though they are common and still supported. The app is built with a version of JavaFX that requires Java 23. On an older Java you get an error like `UnsupportedClassVersionError` and no window at all.
>
> **Java 25 (the current LTS) works, and is the easiest choice.** Java 24 and Java 26 work too.

Do you already have it?

Check before installing anything. See Step 3. Many computers already have Java, but often an older version.

Where to get it

Download **Eclipse Temurin** from **https://adoptium.net/temurin/releases/**

It is free, open source, needs no account, and has no licensing conditions.

On that page choose:

Field	Choose
Version	**25 - LTS** (or 26. Anything **23 or newer** works.)
Package Type	**JDK** — see the note below. A JRE runs the calculator too, but switches one feature off.
Operating System	your OS
Architecture	see below

> **Why JDK and not JRE?** Both run the calculator, and every calculation you type gives the same answer either way. The difference is **speed**. The app can turn your own functions into machine code. That is `compile()`, and an automatic version of it runs once a function has been working hard. Both need the Java *compiler*, and only the JDK has one.
>
> On a JRE, `compile()` gives an error explaining that this Java has no compiler. The automatic version is harder to notice: it does not run at all, and there is no message on screen. The JDK is a somewhat larger download and costs you nothing else, so take it.
>
> Already installed a JRE? Nothing is broken and nothing needs uninstalling, and the calculator works. Install the JDK whenever you like, and the feature becomes available.

Architecture. This is the field people most often get wrong:

Your computer	Architecture
Windows	**x64**
Intel Mac	**x64**
Apple silicon Mac (M1–M4)	**aarch64**
Linux (most)	**x64**

Download the **installer** (`.msi` on Windows, `.pkg` on macOS) rather than the `.zip` / `.tar.gz`. The installer sets everything up for you.

Install it

Windows. Run the downloaded `.msi` and click through. On the options screen, make sure **"Set JAVA_HOME variable"** and **"Add to PATH"** are enabled (they usually are by default).

macOS. Run the downloaded `.pkg` and click through. There is nothing to configure.

Linux. You can use the `.tar.gz` from Adoptium, but your package manager is easier:

```
# Debian / Ubuntu / Mint / Chromebook Linux
sudo apt update
sudo apt install openjdk-25-jdk        # if not found, try: openjdk-24-jdk

# Fedora
sudo dnf install java-25-openjdk-devel

# Arch
sudo pacman -S jdk-openjdk
```

These are the **JDK** packages, the ones with a compiler in them. The `-jre` / `jre-` spellings install the smaller runtime instead, which runs the calculator but leaves `compile()` unavailable.

If your distribution only offers Java 17 or 21, those are **too old**. Use the Adoptium download instead.

After installing, close every terminal window and open a new one. A terminal only sees the programs that existed when it started.

Step 3 — Check your Java version

This step is not optional. Without it you do not know whether step 2 worked.

Windows. Open **PowerShell** (Start menu → type `powershell`):

macOS. Open **Terminal** (Applications → Utilities → Terminal, or Cmd+Space → "Terminal"):

Linux. Open your terminal:

Then in all three, type:

```
java -version
```

What you should see

```
openjdk version "25.0.3" 2026-01-20
OpenJDK Runtime Environment Temurin-25.0.3+9 (build 25.0.3+9)
OpenJDK 64-Bit Server VM Temurin-25.0.3+9 (build 25.0.3+9, mixed mode)
```

Look at the first number. It must be **23 or higher**.

What you see	Meaning
`version "25.0.3"` or `"26…"` or `"23…"`	✅ good
`version "21.0.5"` / `"17.0.9"` / `"11…"` / `"1.8…"`	❌ **too old** — go back to step 2
`'java' is not recognized…` (Windows)	❌ Java not installed, or terminal opened before installing
`command not found: java` (macOS/Linux)	❌ same

> **"But I just installed it!"** Close the terminal and open a **new** one. This catches almost everyone.

> **Careful:** if you had an old Java already, `java -version` may still show the old one. In that case the new install did not take priority. On Windows, uninstall the old Java from *Settings → Apps*. Ask your instructor if unsure.

And check for the compiler

`java -version` cannot tell you whether you got the JDK or the JRE, because both print exactly the same thing. This command answers it:

```
javac -version
```

What you see	Meaning
`javac 25.0.3` (any version 23 or higher)	✅ you have the JDK — `compile()` will work
`'javac' is not recognized` / `command not found: javac`	you have the JRE. The calculator runs; `compile()` and the automatic speed-up do not. See step 2

This one is not pass-or-fail. A missing `javac` costs you speed on long calculations and nothing else, and every answer is identical either way.

Step 4 — Run the calculator

Put the jar somewhere you can find it. Your Desktop is fine.

Do not double-click the jar. It sometimes works, but usually one of these happens instead:

- **Windows:** a zip program (7-Zip, WinRAR) opens it as an archive
- **macOS:** the security system blocks it
- **Linux:** nothing happens at all

Use a terminal instead. It always works, and it shows any error message.

Windows (PowerShell)

```
cd $HOME\Desktop
java -XX:MaxRAMPercentage=50 -jar ScientificCalculator-win-linux-intelmac.jar
```

macOS / Linux (Terminal)

```
cd ~/Desktop
java -XX:MaxRAMPercentage=50 -jar ScientificCalculator-mac-aarch64.jar
```

(Use whichever filename you were given; see step 1.)

The window should open in a few seconds. **Leave the terminal open** while you use the calculator. Closing it closes the app.

What is `-XX:MaxRAMPercentage=50`?

It gives the calculator **half your computer's memory** to work with. Include it: without it the calculator takes only about a quarter, and a calculation that works on your friend's laptop may run out of memory on yours. The program itself is small, and it opens with every tab built on under 50 MB. The flag is there for the room your own calculations get. It also sets how large a saved workspace the calculator will open: a workspace bigger than a quarter of that memory will not open, and the message gives both figures, so a huge workspace saved on a big computer may need the same flag, or more memory, to open on a small one.

You can confirm it worked: the bottom of the window shows `Memory` followed by the figure you actually got. Roughly half your RAM means the flag took effect:

Your computer's RAM	You should see about
4 GB	`Memory 2048 MB`
8 GB	`Memory 4096 MB`
16 GB	`Memory 8192 MB`
32 GB	`Memory 16384 MB`

If it shows roughly a **quarter** of your RAM instead, the flag was left out or mistyped.

> There is no single number to check here, because the flag asks for a *share* of your machine instead of a fixed amount. A fixed figure would have to be small enough for the smallest machine in the class, and image work needs room. Everyone gets the same share, and the status bar shows the figure you actually got.
>
> If you would prefer one exact figure on every computer, type `-Xmx4g` instead. That asks for 4096 MB no matter what the machine has. Use a smaller number if your computer has 4 GB or less.

Step 5 — You're in. Start with the User Manual tab

The window has sixteen tabs in both editions. In Basic, nine of them open as a page describing what that tab does in the Advanced Edition, so the tabs stay in the same places and this guide reads the same either way. **Standard Calc** is where you type. **Accessibility**, second from the left, is a map of the whole program and the fastest way to get anywhere without a mouse: press **Ctrl+G** from anywhere, type a few letters of what you want, and press Enter. **User Manual** is a complete guide that lives inside the app, so you never have to hunt for documentation elsewhere. **About**, on the far right, explains why the program was written and how to reach its author. Please do write.

If you have never used it before, open **User Manual** and read from the top. It opens with *Start Here* (what the tabs are, your first calculation) followed by *First Steps*, four complete tasks worked through end to end. Every example in it can be typed in exactly as printed.

Two things to know on day one:

- The **DEG / RAD** button at the top-left sets whether `sin(90)` means 1 or 0.894. Both are correct; you have to know what they mean, sine of 90 degrees is 1 and sine of 90 radians is 0.894. Check the green label before trusting any trig result.
- Nothing you type can break anything. A mistake comes back as a message in `[square brackets]` and your work survives. Experiment freely.

Making a shortcut (optional)

This saves typing the command each time.

Windows. Make a file called `run-calculator.bat` next to the jar, containing:

```
@echo off
java -XX:MaxRAMPercentage=50 -jar "%~dp0ScientificCalculator-win-linux-intelmac.jar"
pause
```

Double-click that instead. (`%~dp0` means "the folder this file is in", so it works wherever you put it.) The `pause` keeps the window open so you can read any error message.

macOS / Linux. Make a file called `run-calculator.sh` next to the jar, containing:

```
#!/usr/bin/env bash
cd "$(dirname "$0")"
java -XX:MaxRAMPercentage=50 -jar ScientificCalculator-mac-aarch64.jar
```

Then once, in the terminal:

```
chmod +x run-calculator.sh
```

If it doesn't work

What you see	What it means	Fix
`'java' is not recognized` / `command not found: java`	Java isn't installed, or the terminal was opened before installing	Step 2, then open a **new** terminal
`UnsupportedClassVersionError` … `class file version 67.0`	**Your Java is too old.** This is the most common failure.	Install Java 23+ (step 2)
`Error: Unable to access jarfile …`	You're in the wrong folder, or the filename is mistyped	`cd` to where the jar is; check the name exactly
`no suitable image found` / `mach-o` / `libglass.dylib` errors on a Mac	**You have the wrong Mac file**	Step 1 — check Intel vs Apple silicon
`JavaFX runtime components are missing`	The jar is damaged or incomplete	Ask for a fresh copy — <info@srivus.com>
Window opens but `Memory` shows about a QUARTER of your RAM	You left out `-XX:MaxRAMPercentage=50`	Add it
`compile(...)` answers "*this copy of Java does not have one (it is a JRE, not a JDK)*"	You have the JRE, not the JDK. Not a fault — the calculator is working	Install the JDK (step 2). Everything keeps working meanwhile, just not as fast
Nothing happens on double-click	Expected — see step 4	Use the terminal

Reporting a problem

Include all of this, so the first reply can be an answer:

1. Your OS and, on a Mac, whether it is Intel or Apple silicon
2. The **complete** output of `java -version`
3. Which jar file you are using, and its exact filename
4. The **exact** command you typed
5. The **complete** error message, copied as text

Quick reference

```
1. Which file?    Apple silicon Mac -> ScientificCalculator-mac-aarch64.jar
                  everything else   -> ScientificCalculator-win-linux-intelmac.jar

2. Get Java 23+   https://adoptium.net/temurin/releases/   (Temurin 25 LTS, JDK)

3. Check it       java -version        -> first number must be 23 or higher
                  javac -version       -> JDK if present; a JRE has no compile()

4. Run it         java -XX:MaxRAMPercentage=50 -jar <the file you chose>

5. Confirm        status bar "Memory" shows about HALF your computer's RAM

6. Learn it       open the User Manual tab -> Start Here -> First Steps
```