

Java version mismatch error



```
explain following error why it cause and how to resolve this will occu /usr/bin/env
/usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp
/home/parmanand/.config/Code/User/workspaceStorage/f9e76485f007fb23d58fd52136c871ef/
redhat.java/jdt_ws/core\ java_176ee0fe/bin practice.ch_3.p1
└──(parmanand㉿devil)-[~/Desktop/core java]
└─$ /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -
XX:+ShowCodeDetailsInExceptionMessages -cp
/home/parmanand/.config/Code/User/workspaceStorage/f9e76485f007fb23d58fd52136c871ef/
redhat.java/jdt_ws/core\ java_176ee0fe/bin practice.ch_3.p1
Error: LinkageError occurred while loading main class practice.ch_3.p1
java.lang.UnsupportedClassVersionError: practice/ch_3/p1 has been compiled by a more
recent version of the Java Runtime (class file version 67.0), this version of the Java Runtime only
recognizes class file versions up to 65.0r
```

Excellent — this is a **common Java version mismatch error**.

Let's break it down carefully.

Error Summary

pgsql

```
Error: LinkageError occurred while loading main class practice.ch_3.p1
java.lang.UnsupportedClassVersionError: practice/ch_3/p1 has been compiled by a more recent
version of the Java Runtime (class file version 67.0), this version of the Java Runtime only
recognizes class file versions up to 65.0
```

What This Means

- The **class file version (67.0)** corresponds to **Java 23**.
- Your **Java Runtime (JRE)** — the one actually running the code — only supports up to **class file version 65.0**, which is **Java 21**.

So:

- You **compiled** your **.class** file using **Java 23 (or a newer compiler)**
- But you are **running** it with **Java 21**

Why It Happens

When you compile a **.java** file, the compiler encodes the Java version it targets into the **.class** file. Each Java version corresponds to a specific *class file version*:

Java Version	Class File Version
Java 17	61.0
Java 18	62.0
Java 19	63.0
Java 20	64.0
Java 21	65.0
Java 22	66.0
Java 23	67.0

If the runtime (JRE) is **older** than the compiler's version, it cannot interpret the newer bytecode format
→ hence this error.

⌚ How to Fix It

You have two main options:

✓ Option 1: Run with a newer JDK/JRE (same or newer than compile version)

If your class was compiled with Java 23:

```
bash
/usr/lib/jvm/java-23-openjdk-amd64/bin/java -cp <your_classpath> practice.ch_3.p1
```

Or install Java 23 (if you don't have it):

```
bash
sudo apt install openjdk-23-jdk
```

Then make it your default:

```
bash
sudo update-alternatives --config java
sudo update-alternatives --config javac
```

✓ Option 2: Recompile targeting an older version (Java 21)

If you want to keep using Java 21 at runtime, recompile your code to target Java 21:

```
bash
javac --release 21 practice/ch_3/p1.java
```

Or explicitly set source/target:

```
bash
javac -source 21 -target 21 practice/ch_3/p1.java
```

Then rerun:

```
bash
java -cp . practice.ch_3.p1
```

TL;DR

Cause	Fix
Class compiled with newer JDK than runtime supports	Either run with newer JRE (Java 23) or recompile for Java 21

Would you like me to show how to check exactly which JDK version your VS Code or system compiler is using (so this mismatch doesn't happen again)?



```
i have this —(parmanand@devil)-[~/Desktop/core java]
└$ java --version
openjdk 21.0.9-ea 2025-10-21
OpenJDK Runtime Environment (build 21.0.9-ea+8-Debian-1)
OpenJDK 64-Bit Server VM (build 21.0.9-ea+8-Debian-1, mixed mode, sharing)
```

Perfect — thanks for checking that.

So here's what's happening in your case 🤔

>The Situation

You're running:

```
nginx
openjdk 21.0.9-ea
```

→ That means your **Java Runtime (JRE)** is **Java 21**, which supports **class file version 65.0**.

But the error says:

```
pgsql
has been compiled by a more recent version of the Java Runtime (class file version 67.0)
```

→ That means your **.class** file was compiled using **Java 23** (class version 67.0).

So your system **is running Java 21**,
but **the compiled bytecode was produced by Java 23 (or newer)**.

Why That Happens

This mismatch usually happens in **VS Code** (or IntelliJ, Eclipse, etc.) when:

- The **Java extension** (e.g. Red Hat Java) uses a **different JDK** to compile code than your terminal's `java` command.
- For example, your VS Code might be configured to use `/usr/lib/jvm/java-23-openjdk...` for building projects,
but your terminal command uses `/usr/lib/jvm/java-21-openjdk...` to run them.

So the code is compiled by **Java 23**, but you're trying to run it with **Java 21**.

How to Fix It

You have **two clean options**:

Option 1 — Run the program with Java 23

Install or switch to Java 23 if you want to keep the compiler settings as-is:

```
bash
sudo apt install openjdk-23-jdk
```

Then verify:

```
bash
java --version
```

And if needed, make it the default:

```
bash
sudo update-alternatives --config java
sudo update-alternatives --config javac
```

Now rerun your command — it should work fine.

Option 2 — Recompile using Java 21 (recommended if you only have 21)

In your terminal (where `javac --version` should also show Java 21):

```
bash
javac --release 21 /path/to/practice/ch_3/p1.java
```

or if you're inside the source directory:

```
bash
javac --release 21 practice/ch_3/p1.java
```

Then rerun:

```
bash
java practice.ch_3.p1
```

That will compile bytecode compatible with Java 21 (class version 65.0).

Bonus Tip — Fix VS Code's Java Version

If you're using VS Code (you are, based on your path):

```
swift
/home/parmanand/.config/Code/User/workspaceStorage/.../redhat.java/jdt_ws/
```

You can tell VS Code to use Java 21 for both **compiling** and **running**:

1. Open **Command Palette** → “Preferences: Open Settings (JSON)”
2. Add (or modify) this setting:

```
json
"java.configuration.runtimes": [
  {
    "name": "JavaSE-21",
    "path": "/usr/lib/jvm/java-21-openjdk-amd64",
    "default": true
  }
]
```

Then reload VS Code.

Summary

Problem	Fix
Compiled with Java 23 (class file 67.0)	Run with Java 23 OR recompile with --release 21
VS Code uses newer JDK than terminal	Set "java.configuration.runtimes" to point to JDK 21

Would you like me to show the exact command to find **which JDK VS Code is using to compile** right now? It'll confirm whether that's the mismatch.



```
(parmanand@devil)-[~/Desktop/core java]
$ sudo apt install openjdk-23-jdk
```

Error: Unable to locate package openjdk-23-jdk

Good — this tells us you're on a Debian/Ubuntu-based system, and that **OpenJDK 23 isn't yet in your distro's default repositories** (which is common in 2025 — many Linux repos lag one or two releases behind Oracle's or Adoptium's latest JDKs).

You still have **three solid ways** to get Java 23.

Option 1: Use Adoptium's (Temurin) packages — easiest and official

Temurin (Adoptium) provides up-to-date OpenJDK builds for all major versions.

Steps

```
bash

sudo apt update
sudo apt install wget apt-transport-https gpg
wget -O- https://packages.adoptium.net/artifactory/api/gpg/key/public | sudo gpg --dearmor -o
/usr/share/keyrings/adoptium.gpg

# Add the repo
echo "deb [signed-by=/usr/share/keyrings/adoptium.gpg]
https://packages.adoptium.net/artifactory/deb $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/adoptium.list

sudo apt update
sudo apt install temurin-23-jdk
```

Then check:

```
bash

java --version
```

If you see **openjdk 23** or **Temurin-23**, you're good.

Option 2: Use SDKMAN! (great for switching multiple JDKs)

If you like flexibility (for switching between 21 ↔ 23 ↔ 17 easily):

```
bash

curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk install java 23-open
sdk use java 23-open
```

Then verify:

```
bash

java --version
```

Option 3: Download manually (if you prefer direct install)

Go to <https://adoptium.net/temurin/releases/?version=23>

Download the Linux x64 tarball, then:

```
bash

cd ~/Downloads
tar -xvf OpenJDK23U-jdk_x64_linux_*tar.gz
sudo mv jdk-23* /usr/lib/jvm/java-23-openjdk
sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/java-23-openjdk/bin/java 2300
sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/java-23-openjdk/bin/javac 2300
sudo update-alternatives --config java
sudo update-alternatives --config javac
```

After installation

Confirm:

```
bash

java --version
javac --version
```

Then rerun your class:

```
bash

java -cp <your_classpath> practice.ch_3.p1
```

Would you like me to tell you **which VS Code setting to point to this new JDK 23 once it's installed**, so your build and run both stay in sync?



```

└─(parmanand@devil)-[~/Desktop/core java]
└─$ sudo apt update
sudo apt install wget apt-transport-https gpg
wget -O- https://packages.adoptium.net/artifactory/api/gpg/key/public | sudo gpg --dearmor -o
/usr/share/keyrings/adoptium.gpg

# Add the repo
echo "deb [signed-by=/usr/share/keyrings/adoptium.gpg]
https://packages.adoptium.net/artifactory/deb $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/adoptium.list

sudo apt update
sudo apt install temurin-23-jdk

Hit:1 http://http.kali.org/kali kali-rolling InRelease
Hit:2 https://packages.microsoft.com/repos/code stable InRelease
Hit:3 https://brave-browser-apt-release.s3.brave.com stable InRelease
Hit:4 https://repo.protonvpn.com/debian stable InRelease
Ign:5 https://packages.adoptium.net/artifactory/deb kali-rolling InRelease
Err:6 https://packages.adoptium.net/artifactory/deb kali-rolling Release
  404 Not Found [IP: 104.18.21.66 443]
Error: The repository 'https://packages.adoptium.net/artifactory/deb kali-rolling Release' does
not have a Release file.
Notice: Updating from such a repository can't be done securely, and is therefore disabled by
default.
Notice: See apt-secure(8) manpage for repository creation and user configuration details.
wget is already the newest version (1.25.0-2).
apt-transport-https is already the newest version (3.1.8+kali1).
gpg is already the newest version (2.4.8-4).
gpg set to manually installed.
Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 508
--2025-11-10 12:37:13- https://packages.adoptium.net/artifactory/api/gpg/key/public
Resolving packages.adoptium.net (packages.adoptium.net)... File
'/usr/share/keyrings/adoptium.gpg' exists. Overwrite? (y/N) 104.18.20.66, 104.18.21.66,
2606:4700:8d92:c6ae:52d9:549:b43a:4f7c
          Connecting to packages.adoptium.net
(packages.adoptium.net)|104.18.20.66|:443... connected.                                HTTP request sent,
awaiting response... 200 OK
Length: 1793 (1.8K) [text/plain]
          Saving to: 'STDOUT'

-
  100%
[=====>] 1.75K -
-.KB/s  in 0s
2025-11-10
12:37:13 (24.8 MB/s) - written to stdout [1793/1793]

```

Enter new filename:

```
gpg: no valid OpenPGP data found.
gpg: dearmoring failed: File exists
deb [signed-by=/usr/share/keyrings/adoptium.gpg]
https://packages.adoptium.net/artifactory/deb kali-rolling main
Hit:1 https://brave-browser-apt-release.s3.brave.com stable InRelease
Hit:2 http://http.kali.org/kali kali-rolling InRelease
Ign:3 https://packages.adoptium.net/artifactory/deb kali-rolling InRelease
Hit:4 https://packages.microsoft.com/repos/code stable InRelease
Err:5 https://packages.adoptium.net/artifactory/deb kali-rolling Release
  404  Not Found [IP: 104.18.20.66 443]
Hit:6 https://repo.protonvpn.com/debian stable InRelease
Error: The repository 'https://packages.adoptium.net/artifactory/deb kali-rolling Release' does
not have a Release file.
Notice: Updating from such a repository can't be done securely, and is therefore disabled by
default.
Notice: See apt-secure(8) manpage for repository creation and user configuration details.
Error: Unable to locate package temurin-23-jdk
```

```
└─(parmanand㉿devil)-[~/Desktop/core java]
└$
```

Perfect — this output explains *exactly* what's happening 🤦

💡 Why it failed

Adoptium's repository **doesn't provide a `kali-rolling` build target**, so when apt tries to fetch

```
arduino

https://packages.adoptium.net/artifactory/deb kali-rolling Release
```

it gets a **404 (not found)**.

The repo works for **Ubuntu/Debian releases** like:

- `jammy` (Ubuntu 22.04)
- `noble` (Ubuntu 24.04)
- `bookworm` (Debian 12)

but **not for Kali Linux**, since Kali's codename (`kali-rolling`) isn't supported directly.

✓ Solution: Use Debian “bookworm” as the release name

You can safely tell apt to use the `bookworm` repo instead — it works fine on Kali.

Run these commands step-by-step:

```
bash

# Remove the old repo entry
sudo rm /etc/apt/sources.list.d/adoptium.list

# Re-add the repo using Debian bookworm instead of kali-rolling
echo "deb [signed-by=/usr/share/keyrings/adoptium.gpg]
https://packages.adoptium.net/artifactory/deb bookworm main" | \
sudo tee /etc/apt/sources.list.d/adoptium.list

# Update & install
sudo apt update
sudo apt install temurin-23-jdk
```

This will correctly fetch the packages.

Then verify installation

```
bash

java --version
javac --version
```

You should now see something like:

```
yaml

openjdk 23.0.x 2025-xx-xx
OpenJDK Runtime Environment Temurin-23+...
```

Optional: Set Java 23 as default

```
bash

sudo update-alternatives --config java
sudo update-alternatives --config javac
```

Pick the Java 23 entries.

Tip

If that still fails for some reason (rare on Kali), the **SDKMAN!** method is 100% guaranteed to work, independent of repositories:

```
bash

curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk install java 23-open
sdk use java 23-open
```

Would you like me to walk you through setting VS Code to use the newly installed Java 23 (so it matches your runtime)?



i am kali linux user i want to update my java version

Perfect — since you're on **Kali Linux**, here's a **clear, reliable guide** to update Java properly (and safely) — even though Kali's repos don't yet include the latest JDKs.

Let's go step-by-step

STEP 1 — Check your current version

You already did:

```
bash
java --version
```

It shows:

```
nginx
openjdk 21.0.9-ea ...
```

So you're currently on **Java 21**.

We'll upgrade to **Java 23 (latest stable OpenJDK)**.

STEP 2 — Choose your update method

Because Kali is a **rolling Debian-based distro**, you have **two solid methods**:

METHOD 1: Install via SDKMAN (Recommended for Kali)

SDKMAN is a command-line tool that makes installing and switching between Java versions easy — no repo issues, no manual paths.

1 Install SDKMAN

```
bash
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
```

2 Install the latest OpenJDK 23

```
bash
sdk install java 23-open
```

You can check all available versions:

```
bash
sdk list java
```

Then switch to Java 23:

```
bash
sdk use java 23-open
```

3 Verify

```
bash
java --version
```

 You should see:

```
nginx
openjdk 23.0.x ...
```

And you can easily switch back anytime:

```
bash
sdk use java 21-open
```

METHOD 2: Manual install from Adoptium (if you prefer APT-style install)

Kali's repos don't have Java 23, but we can use **Debian "bookworm"** Adoptium packages.

Run these:

```
bash
sudo apt install wget apt-transport-https gpg -y
# Add Adoptium GPG key
wget -O- https://packages.adoptium.net/artifactory/api/gpg/key/public | \
  sudo gpg --dearmor -o /usr/share/keyrings/adoptium.gpg
# Add Debian "bookworm" repo (works on Kali)
echo "deb [signed-by=/usr/share/keyrings/adoptium.gpg]
https://packages.adoptium.net/artifactory/deb bookworm main" | \
  sudo tee /etc/apt/sources.list.d/adoptium.list
# Update package list
sudo apt update
```

```
# Install Temurin 23 (OpenJDK 23)
sudo apt install temurin-23-jdk -y
```

Then verify:

```
bash

java --version
javac --version
```

STEP 3 — Set Java 23 as the default

After installation:

```
bash

sudo update-alternatives --config java
sudo update-alternatives --config javac
```

Pick the `/usr/lib/jvm/temurin-23-jdk...` entries.

STEP 4 — Confirm globally

Run:

```
bash

java -version
javac -version
echo $JAVA_HOME
```

If `$JAVA_HOME` isn't set, you can add it to your `~/.bashrc`:

```
bash

echo 'export JAVA_HOME=$(dirname $(dirname $(readlink -f $(which java))))' >> ~/.bashrc
source ~/.bashrc
```

Done!

Now you'll have the latest Java 23 on Kali — no version mismatch errors like before.

Would you like me to show how to make **VS Code** automatically use your new Java 23 (so both the compiler and runtime match)?