The Java™ Tutorials

Trail: Getting Started

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

Lesson: Common Problems (and Their Solutions)

Compiler Problems

Common Error Messages on Microsoft Windows Systems

`'javac' is not recognized as an internal or external command, operable program or batch file`

If you receive this error, Windows cannot find the compiler (`javac`).

Here's one way to tell Windows where to find `javac`. Suppose you installed the JDK in `C:\jdk1.8.0`. At the prompt you would type the following command and press Enter:

```
C:\jdk1.8.0\bin\javac HelloWorldApp.java
```

If you choose this option, you'll have to precede your `javac` and `java` commands with `C:\jdk1.8.0\bin\` each time you compile or run a program. To avoid this extra typing, consult the section Updating the PATH variable in the JDK 8 installation instructions.

`Class names, 'HelloWorldApp', are only accepted if annotation processing is explicitly requested`

If you receive this error, you forgot to include the `.java` suffix when compiling the program. Remember, the command is `javac HelloWorldApp.java` not `javac HelloWorldApp`.

Common Error Messages on UNIX Systems

`javac: Command not found`

If you receive this error, UNIX cannot find the compiler, `javac`.

Here's one way to tell UNIX where to find `javac`. Suppose you installed the JDK in `/usr/local/jdk1.8.0`. At the prompt you would type the following command and press Return:

```
/usr/local/jdk1.8.0/javac HelloWorldApp.java
```

Note: If you choose this option, each time you compile or run a program, you'll have to precede your `javac` and `java` commands with `/usr/local/jdk1.8.0/`. To avoid this extra typing, you could add this information to your PATH variable. The steps for doing so will vary depending on which shell you are currently running.

`Class names, 'HelloWorldApp', are only accepted if annotation processing is explicitly requested`

If you receive this error, you forgot to include the `.java` suffix when compiling the program. Remember, the command is `javac HelloWorldApp.java` not `javac HelloWorldApp`.

Syntax Errors (All Platforms)

If you mistype part of a program, the compiler may issue a *syntax* error. The message usually displays the type of the error, the line number where the error was detected, the code on that line, and the position of the error within the code. Here's an error caused by omitting a semicolon (`;`) at the end of a statement:

```
testing.java:14: `;' expected.
System.out.println("Input has " + count + " chars.")
                                                    ^
1 error
```

Sometimes the compiler can't guess your intent and prints a confusing error message or multiple error messages if the error cascades over several lines. For example, the following code snippet omits a semicolon (`;`) from the bold line:

```
while (System.in.read() != -1)
    count++
System.out.println("Input has " + count + " chars.");
```

When processing this code, the compiler issues two error messages:

```
testing.java:13: Invalid type expression.
        count++
                ^
testing.java:14: Invalid declaration.
    System.out.println("Input has " + count + " chars.");
                     ^
2 errors
```

The compiler issues two error messages because after it processes `count++`, the compiler's state indicates that it's in the middle of an expression. Without the semicolon, the compiler has no way of knowing that the statement is complete.

If you see any compiler errors, then your program did not successfully compile, and the compiler did not create a `.class` file. Carefully verify the program, fix any errors that you detect, and try again.

Semantic Errors

In addition to verifying that your program is syntactically correct, the compiler checks for other basic correctness. For example, the compiler warns you each time you use a variable that has not been initialized:

```
testing.java:13: Variable count may not have been initialized.
        count++
        ^
testing.java:14: Variable count may not have been initialized.
    System.out.println("Input has " + count + " chars.");
                                      ^
2 errors
```

Again, your program did not successfully compile, and the compiler did not create a `.class` file. Fix the error and try again.

Runtime Problems

Error Messages on Microsoft Windows Systems

`Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorldApp`

If you receive this error, `java` cannot find your bytecode file, `HelloWorldApp.class`.

One of the places `java` tries to find your `.class` file is your current directory. So if your `.class` file is in `C:\java`, you should change your current directory to that. To change your directory, type the following command at the prompt and press Enter:

```
cd c:\java
```

The prompt should change to `C:\java>`. If you enter `dir` at the prompt, you should see your `.java` and `.class` files. Now enter `java HelloWorldApp` again.

If you still have problems, you might have to change your CLASSPATH variable. To see if this is necessary, try clobbering the classpath with the following command.

```
set CLASSPATH=
```

Now enter `java HelloWorldApp` again. If the program works now, you'll have to change your CLASSPATH variable. To set this variable, consult the Updating the PATH variable section in the JDK 8 installation instructions. The CLASSPATH variable is set in the same manner.

`Could not find or load main class HelloWorldApp.class`

A common mistake made by beginner programmers is to try and run the `java` launcher on the `.class` file that was created by the compiler. For example, you'll get this error if you try to run your program with `java HelloWorldApp.class` instead of `java HelloWorldApp`. Remember, the argument is the *name of the class* that you want to use, *not* the filename.

`Exception in thread "main" java.lang.NoSuchMethodError: main`

The Java VM requires that the class you execute with it have a `main` method at which to begin execution of your application. A Closer Look at the "Hello World!" Application discusses the `main` method in detail.

Error Messages on UNIX Systems

`Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorldApp`

If you receive this error, `java` cannot find your bytecode file, `HelloWorldApp.class`.

One of the places `java` tries to find your bytecode file is your current directory. So, for example, if your bytecode file is in `/home/jdoe/java`, you should change your current directory to that. To change your directory, type the following command at the prompt and press Return:

```
cd /home/jdoe/java
```

If you enter `pwd` at the prompt, you should see `/home/jdoe/java`. If you enter `ls` at the prompt, you should see your `.java` and `.class` files. Now enter `java HelloWorldApp` again.

If you still have problems, you might have to change your CLASSPATH environment variable. To see if this is necessary, try clobbering the classpath with the following command.

```
unset CLASSPATH
```

Now enter `java HelloWorldApp` again. If the program works now, you'll have to change your CLASSPATH variable in the same manner as the PATH variable above.

`Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorldApp/class`

A common mistake made by beginner programmers is to try and run the `java` launcher on the `.class` file that was created by the compiler. For example, you'll get this error if you try to run your program with `java HelloWorldApp.class` instead of `java HelloWorldApp`. Remember, the argument is the *name of the class* that you want to use, *not* the filename.

`Exception in thread "main" java.lang.NoSuchMethodError: main`

The Java VM requires that the class you execute with it have a `main` method at which to begin execution of your application. A Closer Look at the "Hello World!" Application discusses the `main` method in detail.

Applet or Java Web Start Application Is Blocked

If you are running an application through a browser and get security warnings that say the application is blocked, check the following items:

- Verify that the attributes in the JAR file manifest are set correctly for the environment in which the application is running. The Permissions attribute is required. In a NetBeans project, you can open the manifest file from the Files tab of the NetBeans IDE by expanding the project folder and double-clicking `manifest.mf`.
- Verify that the application is signed by a valid certificate and that the certificate is located in the Signer CA keystore.
- If you are running a local applet, set up a web server to use for testing. You can also add your application to the exception site list, which is managed in the Security tab of the Java Control Panel.

Previous page: Previous Lesson
Next page: End of Trail