JDB

java debugger tool

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com


https://www.facebook.com/tutorialspointindia


https://twitter.com/tutorialspoint

About the Tutorial

The Java Debugger, commonly known as jdb, is a useful tool to detect bugs in Java programs. This is a brief tutorial that provides a basic overview of how to use this tool in practice. In addition, the tutorial also covers how to debug a program through stepping, breakpoints, and managing exceptions.

Audience

This tutorial will be quite useful for beginners learning Java as well as programmers and professionals aspiring to make a career in Testing and Analytics using Java.

Prerequisites

Before you start with this tutorial, you need to know basic Java programming.

Disclaimer & Copyright

Table of Contents

1. INTRODUCTION

Debugging is a technical procedure to find and remove bugs or defects in a program and get expected results. Debugging includes testing and monitoring. It is very complex when the subunits of a program are tightly coupled. We can debug a program using the debugger tools that follow the prescribed APIs. A debugger allows you to step through every aspect of a code, inspect all the elements, and remove errors, if any.

Debugging Techniques

There are different kinds of techniques to debug a Java program. The old method of debugging is by using print statements at the end of every segment which will print the trace statements on the console. Take a look at the following code.

```
pubic class Add

{

   public static void main(String ar[])

   {

      int a=ar[0];

      system.out.println("A : " +a);

      int b=ar[1];

      system.out.println("B : " +b);

      int c = a + b;

      system.out.println("C = a + b : " +c);

   }

}
```

Here, we have a program that adds two numbers and prints the output. Notice that at each step, we have introduced a print statement that prints the state of the program on the console. This is the traditional approach to debug a program.

In addition, we have advanced concepts that can be used to debug a program such as:

- stepping,
- breakpoints, and
- exceptions or watchpoints.

Types of Debugging

We can debug a program using various methods:

- Using Java bytecode (compiled version of Java code)
- Using comments inside the programs
- Attaching class to a running program
- Remote debugging
- Debugging on demand
- Optimized code debugging

Java Debuggers

Here are some examples of Java debuggers that are available in the market:

- IDEs such as Eclipse, Netbeans, etc. contain their own debuggers (Visual cafe, Borland, JBuilder)
- Standalone debugger GUIs (such as Jikes, Java platform debugger, and JProbe)
- Command-line debugger (Sun's JDB)
- Notepad or VI driven (stack trace)

This tutorial covers how to use the command-line debugger, **jdb**.

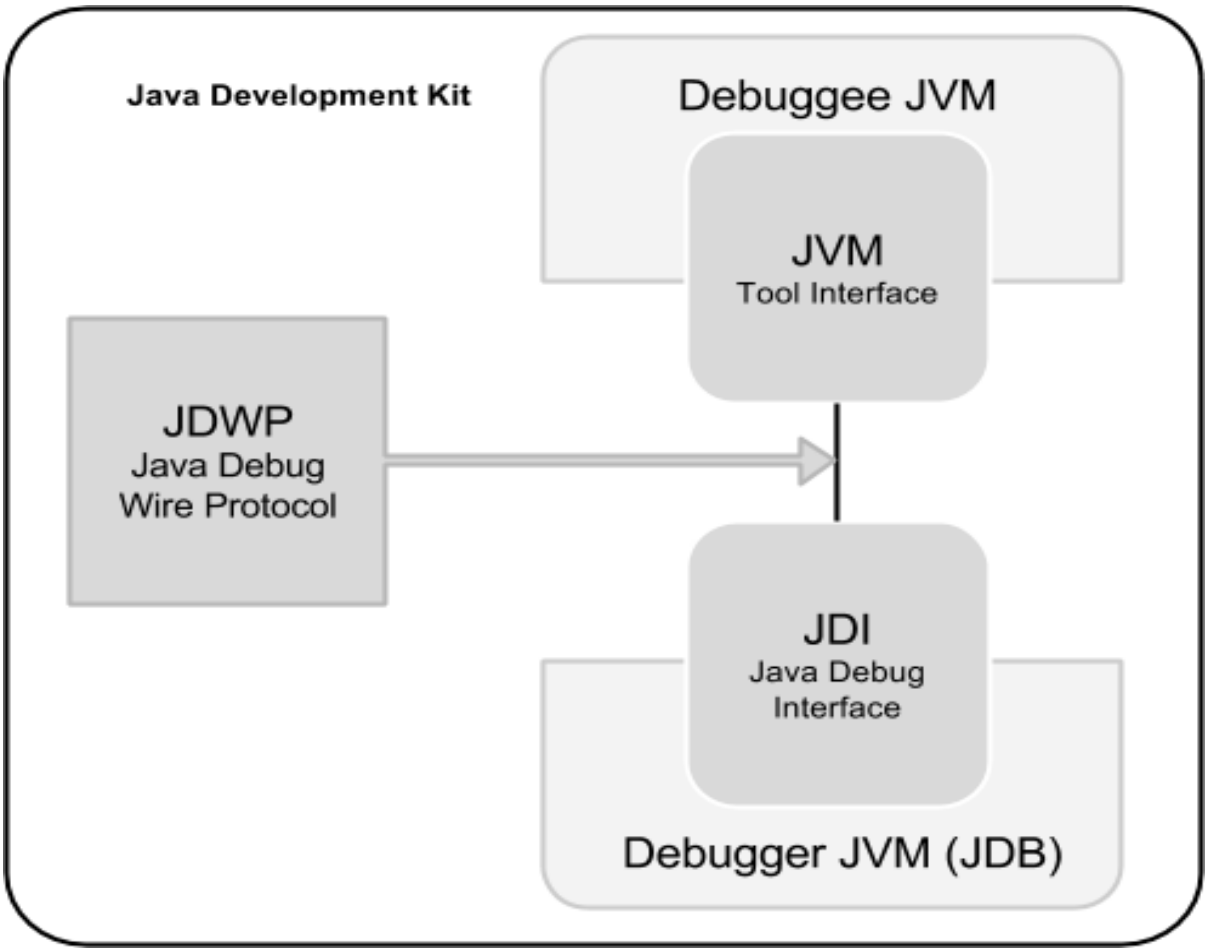
JDB

The Java debugger (JDB) is a tool for Java classes to debug a program in command line. It implements the Java Platform Debugger Architecture. It helps in detecting and fixing bugs in a Java program using Java Debug Interface (JDI).

JDB in JDK

The following architecture defines the role of JDB in JDK. It contains mainly three units:

1. Java Virtual Machine Tool Interface (JVM TI)
2. Java Debug Wiring Pool (JDWP)
3. Java Debugger Interface (JDI)



JVM TI

It is a native programming interface implemented by VM. It provides ways to inspect and debug the state of the application running on the VM. It allows an implementer (VM Implementer) that can be enclosed easily into the debugging architecture. It also uses a third-party channel called **JDWP** for communication.

JDWP

It defines the format of information and the requests that pass in between the debuggee process and the debugger front end. The primary purpose of having a

JDWP is to allow the debuggee and the debugger to communicate when they run under separate VMs or in separate platforms.

JDI

It is a high-level Java interface implemented as front end. It defines the variable information at user code level. It is recommended to use a JDI layer for all debugger development. It uses JDWP for communication with the debuggee JVM.

2. INSTALLATION

This chapter explains how to install JDB on Windows and Linux based systems. JDB is a part of JDK. Therefore, JDK installation is enough for using JDB in command prompt.

System Requirements

Here are the system requirements for installing JDB:

JDK	Java SE 2 JDK 1.5 or above
Memory	1 GB RAM (recommended)
Disk Space	No minimum requirement
Operating System Version	Windows XP or above, Linux

Follow the simple steps given below to install JDB on your system.

Step 1: Verifying Java Installation

First of all, you need to have Java Software Development Kit (SDK) installed on your system. To verify this, execute any of the two commands depending on the platform you are working on.

If the Java installation has been done properly, then it displays the current version and specifications of Java installation. A sample output is given in the following table.

Platform	Command	Sample Output
Windows	Open command console and type: **\>java –version**	Java version "1.7.0_60" Java (TM) SE Run Time Environment (build 1.7.0_60-b19) Java Hotspot (TM) 64-bit Server VM (build 24.60-b09,mixed mode)

Linux	Open command terminal and type: **$java –version**	java version "1.7.0_25" Open JDK Runtime Environment (rhel-2.3.10.4.el6_4-x86_64) Open JDK 64-Bit Server VM (build 23.7-b01, mixed mode)

We assume the readers of this tutorial have Java SDK version 1.7.0_60 installed on their system. In case you do not have Java SDK, download its current version from the link http://www.oracle.com/technetwork/java/javase/downloads/index.html and install it.

Step 2: Setting Up Java Environment

Set the environment variable JAVA_HOME to point to the base directory location where Java is installed on your machine. For example,

Platform	Description
Windows	set JAVA_HOME to C:\ProgramFiles\java\jdk1.7.0_60
Linux	export JAVA_HOME=/usr/local/java

Append the full path of Java compiler location to the System Path.

Platform	Description
Windows	Append the String "C:\Program Files\Java\jdk1.7.0_60\bin" at the end of the system variable PATH.
Linux	export PATH=$PATH:$JAVA_HOME/bin/

Execute the command **java -version** from the command prompt as explained above.

Step 3: Verifying JDB Installation

Verify the JDB version as follows:

Platform	Command	Sample Output
Windows	Open command console and type: **\>jdb –version**	This is JDB version 1.6 (Java SE version 1.7.0_60)
Linux	Open command terminal and type: **$jdb –version**	This is JDB version 1.6 (Java SE version 1.7.0_60)

3. SYNTAX

This chapter explains the syntax of JDB command. The syntax contains four sections listed as follows:

- JDB
- option
- class
- arguments

Syntax

The syntax of JDB is as follows.

```
jdb [ options ] [ class ] [ arguments ]
```

JDB

It calls jdb.exe from the Java Development Kit.

Options

These include the command line options used to debug a Java program in an efficient way. The JDB launcher accepts all the options (such as -D, -classpath, and -X) and some additional advanced options such as (-attach, -listen, -launch, etc.).

Class

It is the class name on which you want to perform debugging operations.

Arguments

These are the input values given to a program at runtime. For example, arg[0], arg[1] to the main() method.

In the above four segments, options is the most important one.