JDB Interview Questions

A list of top frequently asked JDB interview questions and answers are given below:

1) What is a debugger?

A debugger is a computer program used to identify and remove errors from other programs.

2) What is (JBD) Java debugger?

(JDB) Java debugger is a command line Java debugging tool used to debug Java programs without the requirement of inserting specific debugging instructions into the code.

3) What are the advantages of Java Debugger?

These are some advantages of Java Debugger: -

- It is a lightweight debugging tool.
- It is free to use and available for all type of operating systems.
- The execution of Java debugging tool is very fast.
- It provides support for multithreaded programs and remote applications.

4) What are some frequently generate bugs?

These are some commonly generate bugs: -

- Syntactical or Compilation errors - These are errors are generally generated due to some typing mistakes.

- Run-time errors - These errors are generated at execution time generally due to exceptions.
- Threading errors - These errors are difficult to replicate and track down.

5) What are the various types of Java Debugger?

Several types of Java Debuggers are: -

- Stand-alone debuggers such as JLike and JProbe.
- Command-line debugger such as Sun JDB.
- IDEs contain their own debuggers such as Borland JBuilder and IBM VisualAge for Java.

6) What are the different approaches of debugging?

Various types of approaches used for debugging are: -

- Optimized code debugging
- Using comments
- Basic Java bytecode (by using System.out.println())
- Remote Debugging
- Debugging on demand

7) How to connect JDB with JVM?

There are various ways to connect JDB with JVM, but the simplest one is to provide the following command at the runtime of your main class: -

```
>jdb MainClass
```

Here, MainClass represents the name of your class.

8) What is the role of the interpreter in debugging the Java program?

Java debugger interacts with the Java runtime interpreter to interrupt the normal flow of the program. Thus, Java interpreter provides support to the debugger.

9) What is the syntax used to invoke the debugger?

The following syntax is required to invoke debugger: -

jdb [options] [classname] [arguments]

10) How can we debug an applet?

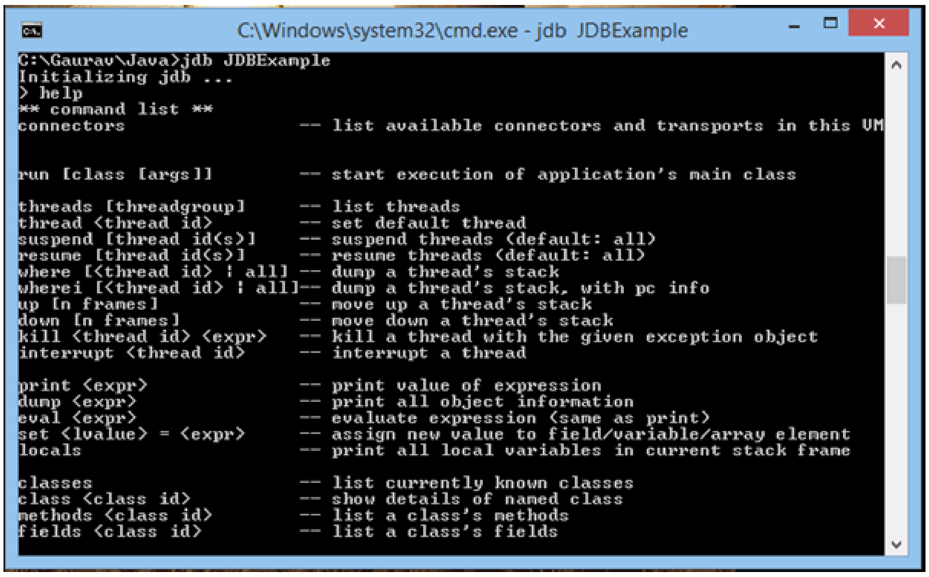
To debug an applet, we have to execute the debugger within applet viewer by the help of the following command: -

```
>appletviewer -
debug URL
```

11) How to list out all the available JDB commands with their specifications?

To list out all the available JDB commands with their specifications, you need to initialize JDB and type **help** command on the console.

After the execution of help command, the following list will be generated on your console: -



```
C:\Gaurav\Java>jdb JDBExample
Initializing jdb ...
> help
** command list **
connectors                    -- list available connectors and transports in this VM


run [class [args]]            -- start execution of application's main class

threads [threadgroup]         -- list threads
thread <thread id>            -- set default thread
suspend [thread id(s)]        -- suspend threads (default: all)
resume [thread id(s)]         -- resume threads (default: all)
where [<thread id> | all]     -- dump a thread's stack
wherei [<thread id> | all]-- dump a thread's stack, with pc info
up [n frames]                 -- move up a thread's stack
down [n frames]               -- move down a thread's stack
kill <thread id> <expr>       -- kill a thread with the given exception object
interrupt <thread id>         -- interrupt a thread

print <expr>                  -- print value of expression
dump <expr>                   -- print all object information
eval <expr>                   -- evaluate expression (same as print)
set <lvalue> = <expr>         -- assign new value to field/variable/array element
locals                        -- print all local variables in current stack frame

classes                       -- list currently known classes
class <class id>              -- show details of named class
methods <class id>            -- list a class's methods
fields <class id>             -- list a class's fields
```

12) How can we start the execution of the main class?

To start the execution of the main class, you need to execute the below command: -

```
>run [class [args]]
```

It is optional to mention the name of specific class and argument.

13) Which command is used to complete the remaining execution?

Use the following command to complete the partial execution of the program.

```
>cont
```

14) What is the difference between print and dump command?

The **print** command is used to display the value of expressions whereas **dump** command is used to display the information of objects.

15) What is the purpose of breakpoints in Java Debugger?

Breakpoints are used in debugging to pause or stop the execution of a program at a specific line of code and test whether the program is functioning correctly or not.

16) What is the syntax used to set a breakpoint?

A breakpoint can be set either on the basis of method name or the specific number of the line.

- To set a breakpoint on the method, the following syntax is used: -

```
stop in
Classname.MethodNa
```

- To set a breakpoint on the specific line, the following syntax is used: -

```
stop at Classname:
LineNumber
```

17) How can we set a breakpoint in an applet?

Let suppose there is an applet named as Javatpoint having a method scrollDown(). To set a breakpoint on scrollDown() method, you need to follow the below command: -

```
>stop in
Javatpoint.scrollD
```

18) What is stepping in Java Debugger?

In JDB, Stepping is a procedure to execute the code line by line. Thus, through this approach, each line of the code can be properly examined.

19) What are the various techniques of stepping?

These are the following techniques of stepping: -

- Step Over
- Step Into
- Step Return

20) Can we handle exceptions through JDB?

Yes, Java debugger is capable of handling runtime exceptions. To handle these exceptions it provides **catch** command.

Java Basics Interview Questions	**Java OOPs Interview Questions**
Java Multithreading Interview Questions	**Java String & Exception Interview Questions**
Java Collection Interview Questions	**JDBC Interview Questions**
Servlet Interview Questions	**JSP Interview Questions**
Spring Interview Questions	**Hibernate Interview Questions**