9.1 Java Debugger (jdb)

The Java debugger may be used to help you figure out what your program is doing just before it triggers an exception. You can use it to run your program normally, run until one or more specified breakpoints is reached, or step through your program line by line. At any point, you can examine the values of the program's variables without needing to insert a print statement and recompile. Properly used, jdb can be a valuable tool in writing bug-free code.

```
[112b@beowulf ~]$ jdb MyProgram
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

threadgroups              -- list threadgroups
threadgroup <name>        -- set current threadgroup

stop in <class id>.<method>[(argument_type,...)]
                          -- set a breakpoint in a method
stop at <class id>:<line> -- set a breakpoint at a line
clear <class id>.<method>[(argument_type,...)]
                          -- clear a breakpoint in a method
clear <class id>:<line>   -- clear a breakpoint at a line
clear                     -- list breakpoints
catch [uncaught|caught|all] <class id>|<class pattern>
                          -- break when specified exception occurs
ignore [uncaught|caught|all] <class id>|<class pattern>
                          -- cancel 'catch' for the specified exception
watch [access|all] <class id>.<field name>
                          -- watch access/modifications to a field
unwatch [access|all] <class id>.<field name>
                          -- discontinue watching access/modifications to a field
trace methods [thread]    -- trace method entry and exit
untrace methods [thread]  -- stop tracing method entry and exit
step                      -- execute current line
step up                   -- execute until the current method returns to its caller
```

```
stepi                      -- execute current instruction
next                       -- step one line (step OVER calls)
cont                       -- continue execution from breakpoint

list [line number|method] -- print source code
use (or sourcepath) [source file path]
                           -- display or change the source path
exclude [<class pattern>, ... | "none"]
                           -- do not report step or method events for specified classes
classpath                  -- print classpath info from target VM

monitor <command>          -- execute command each time the program stops
monitor                    -- list monitors
unmonitor <monitor#>       -- delete a monitor
read <filename>            -- read and execute a command file

lock <expr>                -- print lock info for an object
threadlocks [thread id]    -- print lock info for a thread

pop                        -- pop the stack through and including the current frame
reenter                    -- same as pop, but current frame is reentered
redefine <class id> <class file name>
                           -- redefine the code for a class

disablegc <expr>           -- prevent garbage collection of an object
enablegc <expr>            -- permit garbage collection of an object

!!                         -- repeat last command
<n> <command>              -- repeat command n times
help (or ?)                -- list commands
version                    -- print version information
exit (or quit)             -- exit debugger

<class id>: a full class name with package qualifiers
<class pattern>: a class name with a leading or trailing wildcard ('*')
<thread id>: thread number as reported in the 'threads' command
<expr>: a Java(tm) Programming Language expression.
Most common syntax is supported.

Startup commands can be placed in either "jdb.ini" or ".jdbrc"
in user.home or user.dir
> run
run MyProgram
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
VM Started:
Exception occurred: java.lang.ArrayIndexOutOfBoundsException (uncaught)"thread=AWT-
EventQueue-0", java.awt.EventDispatchThread.run(), line=145 bci=152

AWT-EventQueue-0[1]
```