Stacks:

An abstract data type that helps simplify data management.

Designed to hold collection of elements of the same type.

If stack is not full, you can push a new element onto the stack.

If stack is not empty, you can pop the newest element off the stack.

Elements on the stack are accessed in LIFO order (last in, first out).

Aka: the item most recently put on the stack is the one that will pop off first.

Can be implemented using arrays/nullterminated singly linked list.

Java has a built in Stack class uses a resizable array, based on Vector.

Remember: the prime factorization of n is the unique collection of prime numbers that have n as their product. (ex: prime factorization of 1400 is

1. Compile SampleJDB.java using the -g flag.

javac –g SampleJDB.java

1. Start the program using jdb.
2. Set a breakpoint (stop) when the program enters the lottery method. (The idea here is that this is a method that will be called very rarely -- but you can tell the program to stop when it happens.)
3. Set a breakpoint at line 13. (This should be the line that reads "if (n <= 1) {")
4. Put a watch on SampleJDB.number, so that you will be notified whenever its value changes (but not for ordinary read access).
5. Run the program. (Where does it stop?)
6. print n
7. Continue the program once.
8. print n
9. Clear the breakpoint at SampleJDB:13 and move it to line 14. Continue the program.
10. Show the call stack, with where.
11. print n
12. Go up one stack frame, and print n.
13. Go up one stack frame, and print n again.
14. Continue the program. (Where does it stop next?)
15. Look at the value of SampleJDB.number.
16. Advance the program by one step.
17. Look at the value of SampleJDB.number again.
18. Advance the program by two more step.
19. Continue the program. (Where does it stop next?)
20. Move one step up the call stack, and print out all the locals. (What is i?)

Allow the program to continue and exit.