**Stage 1:**

Java Virtual Machine (JVM) divides the memory into different categories like Stack, Heap, Register, Class, Native method areas to manage memory optimally.

A picture containing chart

Description automatically generated

**Stage 2:**

In Stack area, memory will be allocated for all primitives and references that are part of main() before its execution.

* Primitive type variable “times” of int type will be allocated with memory in the stack directly.
* Reference variable “name” of the String type is created in Stack but its value is stored in “String Pool” which is part of “Heap” area.
* Reference variable “list” of the String type is also created in Stack but points to object located on Heap.
* fill(list, name + name, times), is invoked from main(), which passes a reference from the stack of the main() to the ArrayList object with generic type String, references to the name and list object from the String Pool, and copies the primitive type times.

Diagram

Description automatically generated

**Stage 3:**

Upon invoking the fill(), a new "block" of memory is allocated in the stack on top of main(), following LIFO principle of Stack.

Diagram

Description automatically generated

**Stage 4:**

Upon invoking the shrink(), a new "block" of memory is allocated in the stack on top of fill(), following LIFO principle of Stack.

Diagram

Description automatically generated

**Stage 5:**

After the shrink() complete its execution, JVM frees up the memory area associated with it.

Diagram

Description automatically generated

**Stage 6:**

After the fill() complete its execution, JVM frees up the memory area associated with it.

Diagram, schematic

Description automatically generated

**Stage 7:**

After the main() complete its execution, JVM frees up the memory area associated with it. Program will be terminated and Stack & Heap memory areas will be cleared.

A picture containing chart

Description automatically generated