**How memory works – the stack and the heap**

* **Introduction – the structure of Java’s memory**

**Diagram

Description automatically generated**

* **How the Stack works?**

**Graphical user interface, text, application

Description automatically generated with medium confidence**

**Graphical user interface, application

Description automatically generated**

Once the method is reached the return statement, it pops all the entries related to that method.

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface

Description automatically generated**

Note: The stack is popped off whenever it reaches the return / method end or end of the curly brace of the program.

This is how the scope of the variable are maintained, as and when the method is done, it clears all the variables.

The stack creation is for each thread.

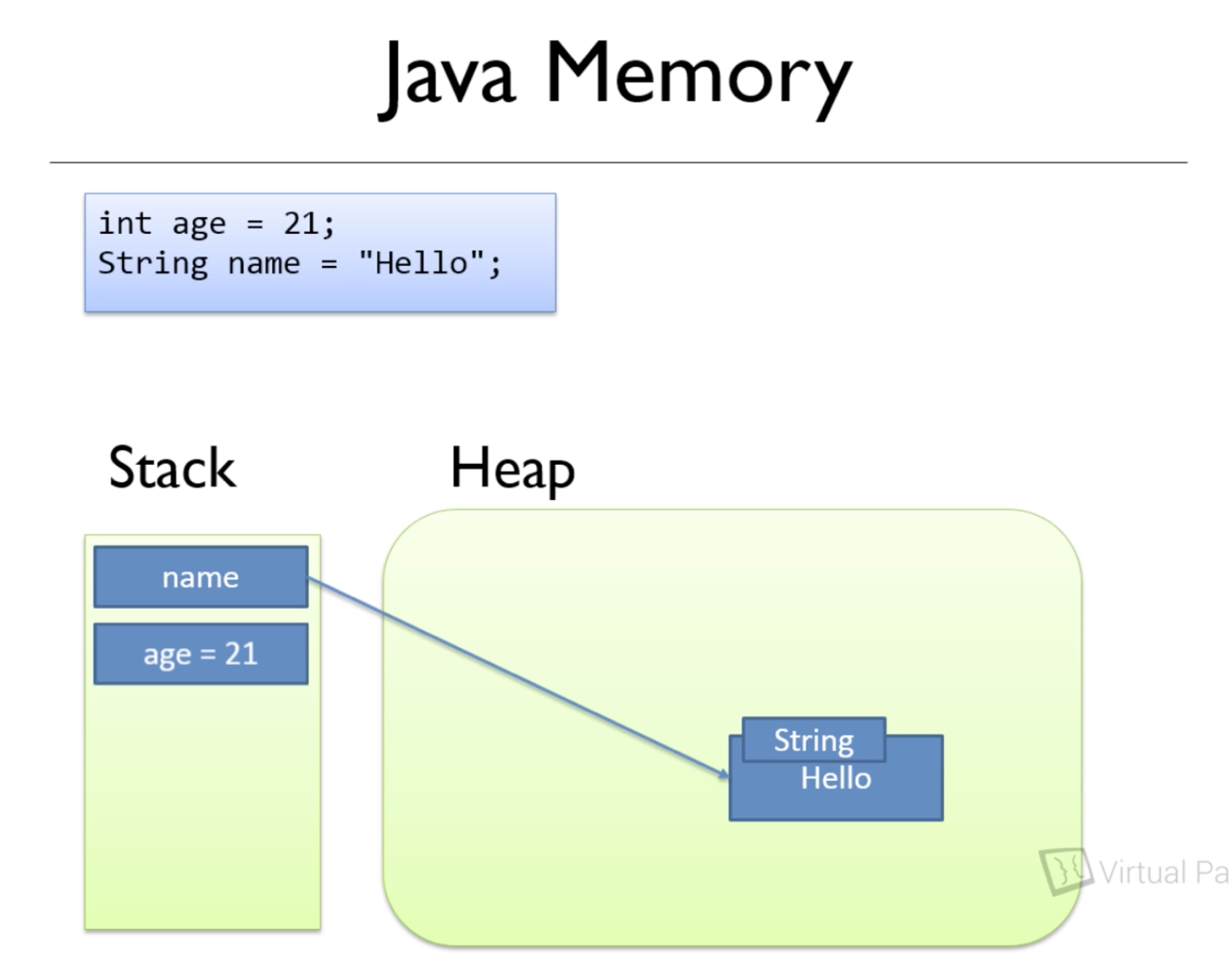
Stack is used for primitive types.

* **How the Heap works?**

Data on the stack can be accessed only by that thread, whereas the heap data is accessed by all the threads.

Heap is used for complex / instance / derived types.



****

****

* **The Heap & the Stack together – an example:**

**Graphical user interface, application

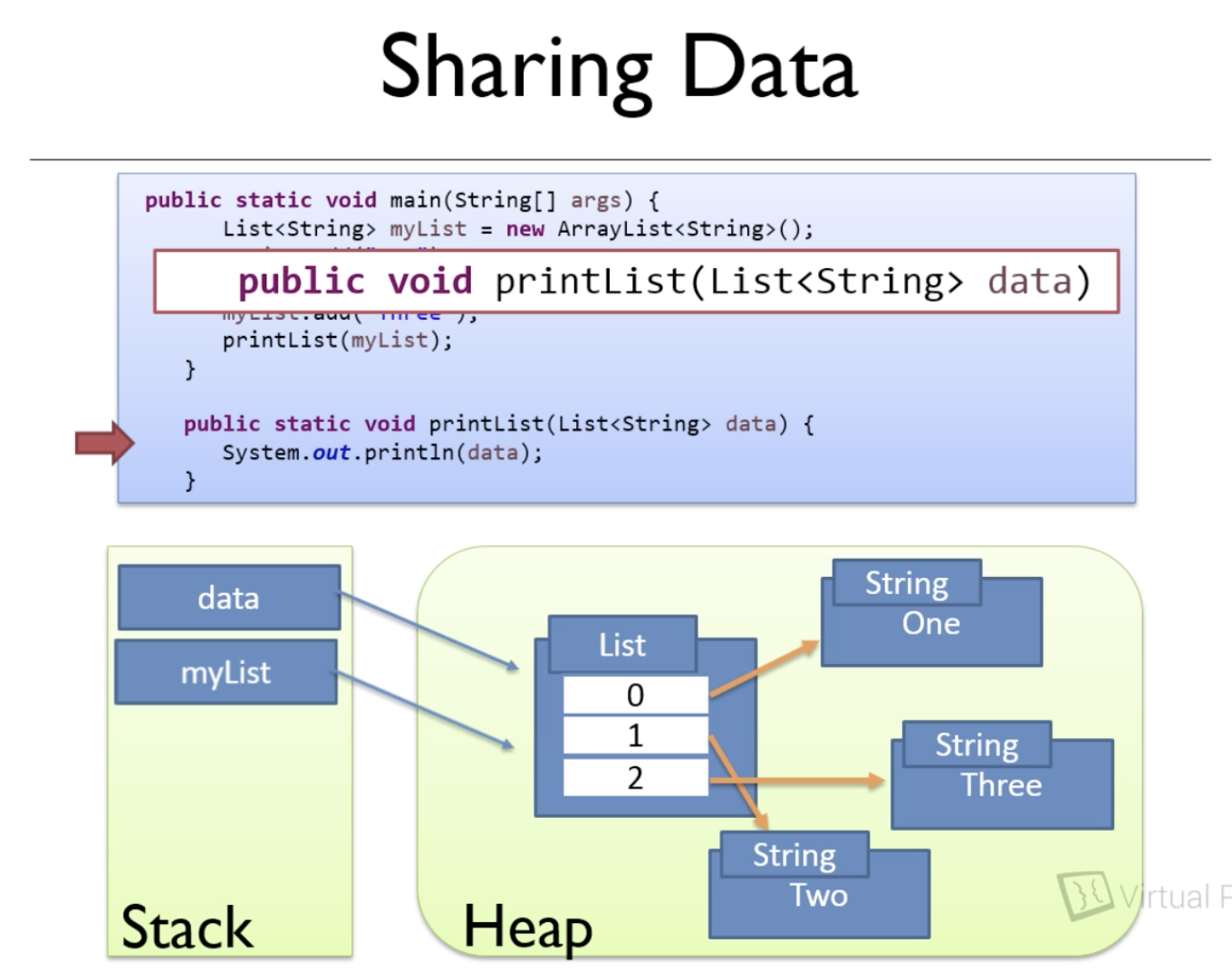
Description automatically generated**

**Graphical user interface, diagram, application

Description automatically generated**

**Graphical user interface, diagram, application

Description automatically generated**

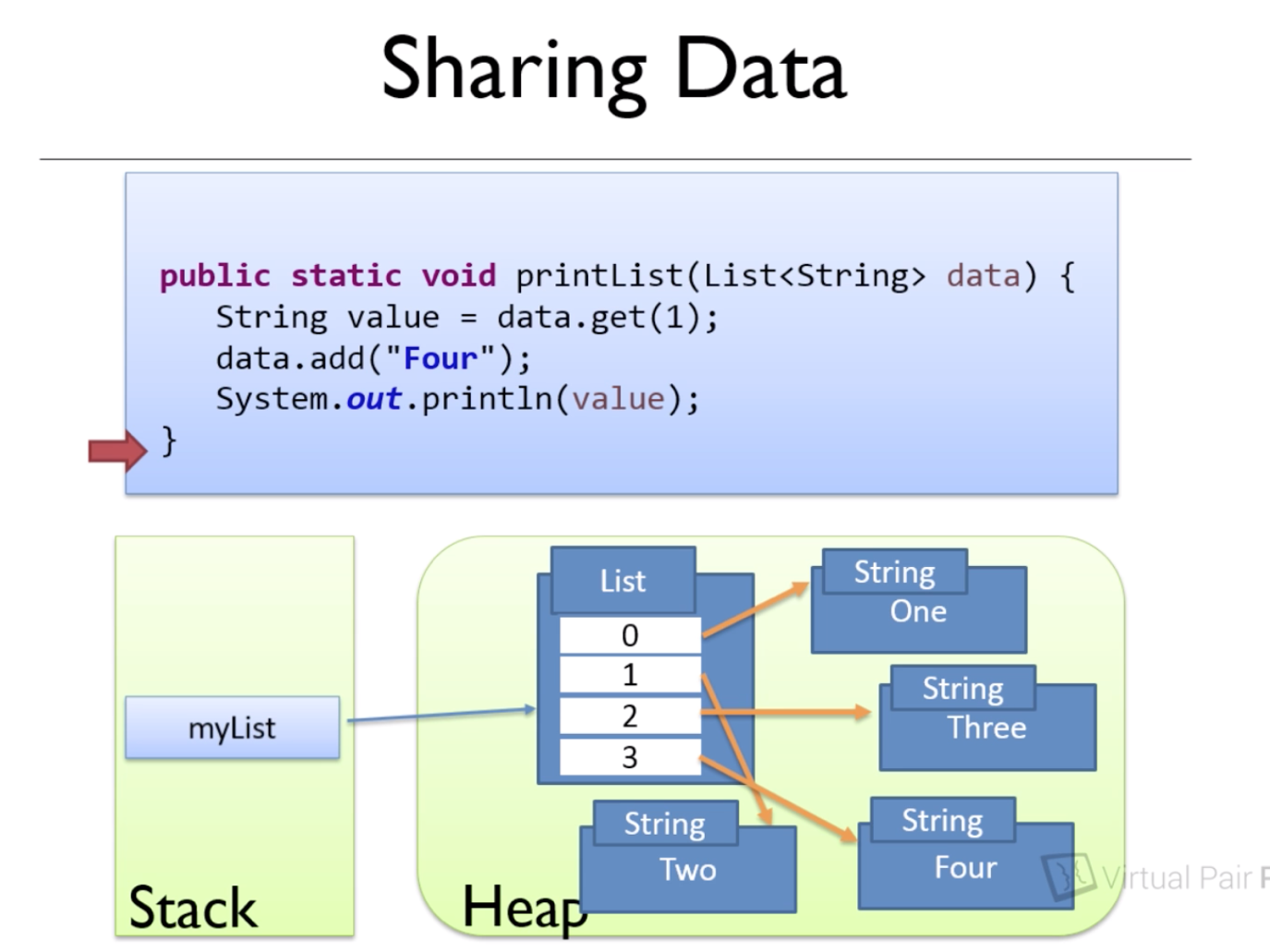
****

**Graphical user interface, diagram

Description automatically generated**

**Diagram

Description automatically generated**

****

**Graphical user interface, diagram

Description automatically generated**