Stack-Heap Diagram

Step 1: `main` Method Entry

Stack:

- `args` (reference to an array of strings passed to `main`)

Step 2: Initializations in `main`

Stack:

- `name` = reference to `"Kevin"` (stored in the heap)
- `list` = reference to a new `ArrayList<>()` (ArrayList object stored in the heap)
- `times` = `10`

Step 3: Evaluate `fill(list, name + name, times)`

Expression Breakdown:

1. `name + name` creates a new `String` object `"KevinKevin"` in the heap.
2. This value is passed as the `str` argument to the `fill` method.

Stack:

- `name` = reference to `"Kevin"`
- `list` = reference to `ArrayList`
- `times` = `10`

Step 4: Call `fill` Method

Stack (inside `fill`):

- `collection` = reference to `list`
- `str` = reference to `"KevinKevin"`
- `times` = `10`

Step 5: Call `shrink(str)` within `fill`

Inside `fill`, call to `shrink` with `str` = `"KevinKevin"`

Step 6: `shrink` Method Execution

1. `newLength` is calculated as `str.length() / 2 + str.length() % 2` → `5`

2. A `char[]` of size 5 is created and populated with characters from `"KevinKevin"` at indices `0`, `2`, `4`, `6`, and `8`, resulting in `{'K', 'v', 'n', 'K', 'v'}`.
3. A new `String` `"KvnKv"` is created from this array.

Stack (inside `shrink`):

- `str` = reference to `"KevinKevin"`
- `newLength` = `5`
- `chars` = `{'K', 'v', 'n', 'K', 'v'}`

Return `"KvnKv"` from `shrink`.

Step 7: `fill` Method Execution (continued)

1. `shrunk` is set to reference `"KvnKv"`.
2. `times` is recalculated as `(10 + "KvnKv".length()) / 2` → `(10 + 5) / 2 = 7`.
3. A loop runs `times / 2 = 7 / 2 = 3` times, adding `"KvnKv"` to `collection` (`list`) three times.

Stack (inside `fill` after `shrink`):

- `collection` = reference to `list`
- `str` = reference to `"KevinKevin"`
- `shrunk` = reference to `"KvnKv"`
- `times` = `7`
- Loop index variable `i` iterates from `0` to `2`.

Heap (state of `list`):

- `list` = `["KvnKv", "KvnKv", "KvnKv"]`

Return `times = 7` from `fill`.

Step 8: `main` Method Execution (continued)

1. `System.out.println(times + fill(list, name + name, times))` evaluates to `10 + 7 = 17`.
2. `17` is printed.

Final State:

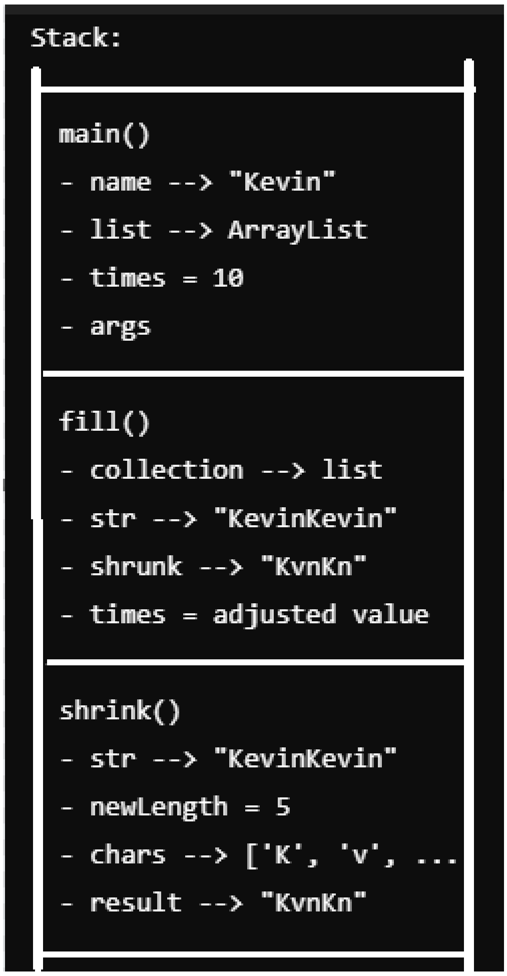
Stack (after `main` completes):

- `name` = reference to `"Kevin"`

- `list` = reference to `ArrayList` containing `["KvnKv", "KvnKv", "KvnKv"]`
- `times` = `10`

Heap:

- `"Kevin"`
- `"KevinKevin"`
- `"KvnKv"`
- `ArrayList` instance (referenced by `list` in `main`), containing `["KvnKv", "KvnKv", "KvnKv"]`

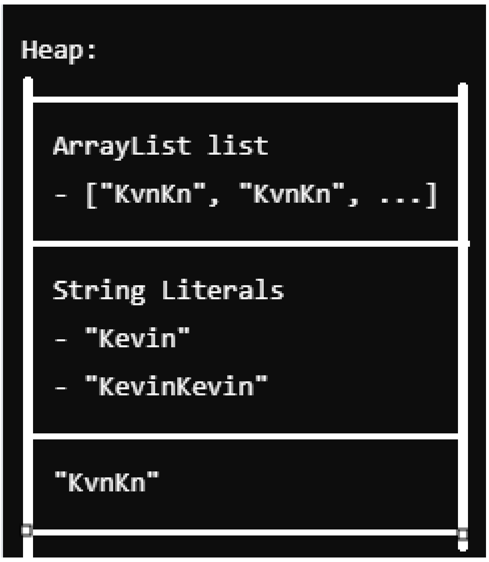


```
Stack:

main()
- name --> "Kevin"
- list --> ArrayList
- times = 10
- args

fill()
- collection --> list
- str --> "KevinKevin"
- shrunk --> "KvnKn"
- times = adjusted value

shrink()
- str --> "KevinKevin"
- newLength = 5
- chars --> ['K', 'v', ...
- result --> "KvnKn"
```



```
Heap:

ArrayList list
- ["KvnKn", "KvnKn", ...]

String Literals
- "Kevin"
- "KevinKevin"

"KvnKn"
```