Well, the `ternary operator` in Java acts like this...

```
return_value = (true-false condition) ? (if true expression) : (if false expression);
```

...Another way of looking at it...

```
return_value = (true-false condition)
             ? (if true expression)
             : (if false expression);
```

You question is kind of vague and we have to assume here.

- If **(and only if)** `callFunction(...)` declares a `non-void` return value (`Object`, `String`, `int`, `double`, etc..) - *it seems like it does not do that via your code* - then you could do this...

```
return_value = (string != null)
             ? (callFunction(...))
             : (null);
```

- If `callFunction(...)` does not return a value, then you **cannot** use the ternary operator! Simple as that. You will be using something that you don't need.
  - Please post more code to clear up any issues

Nonetheless, **ternary operator should represent alternative assignments only!!** Your code does not seem to do that, so you should not be doing that.
This is how they should work...

```
if (obj != null) {            // If-else statement

    retVal = obj.getValue();  // One alternative assignment for retVal

} else {

    retVal = "";              // Second alternative assignment for retVale

}
```

This can be converted to...

```
retVal = (obj != null)
       ? (obj.getValue())
       : ("");
```

Since it seems like you *might* be trying to just refactor this code to be a one-liner, I have added the following

Also, if your false-clause is truely empty, then you can do this...

```
if (string != null) {

    callFunction(...);

} // Take note that there is not false clause because it isn't needed
```

OR

```
if (string != null) callFunction(...);  // One-liner
```