In the builder pattern, method chaining allows for a fluent and expressive interface when configuring object properties. Each method in the chain returns a reference to the builder itself (**this**), enabling consecutive method calls to be chained together. This results in a concise and readable syntax for configuring objects with multiple properties.

Let's examine how method chaining is utilized in the provided code:

javaCopy code

TestRequestBuilderClass builder = new TestRequestBuilderClass.Builder()

.setBaseURI("https://jsonplaceholder.typicode.com/") // Set base URI

.setQueryParam("userid", 1) // Set query parameter 'userid=1'

.build(); // Build the TestRequestBuilderClass instance

In this example:

1. The **Builder()** constructor instantiates a new **Builder** object.
2. The **setBaseURI("https://jsonplaceholder.typicode.com/")** method is invoked on the **Builder** object, setting the base URI for the request. This method returns the same **Builder** object (**this**).
3. The **setQueryParam("userid", 1)** method is called on the same **Builder** object, setting the query parameter "userid" to 1. Again, this method returns the **Builder** object.
4. Finally, the **build()** method is invoked, which constructs and returns an instance of **TestRequestBuilderClass** configured with the properties set using the builder methods.

By chaining these method calls together, the code achieves a concise and readable representation of the object configuration process. Each method call is descriptive and self-contained, contributing to the overall clarity and maintainability of the code.

Method chaining is a powerful technique that simplifies object construction and configuration, making the builder pattern an effective solution for creating complex objects with optional parameters. It enhances code readability, reduces verbosity, and promotes a fluent interface for interacting with objects.