HBASE:

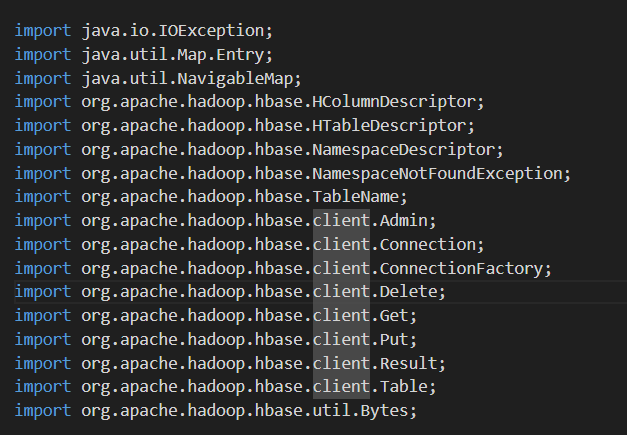
1. store as HelloHBase.java

2. javac -cp $(hbase classpath) HelloHBase.java

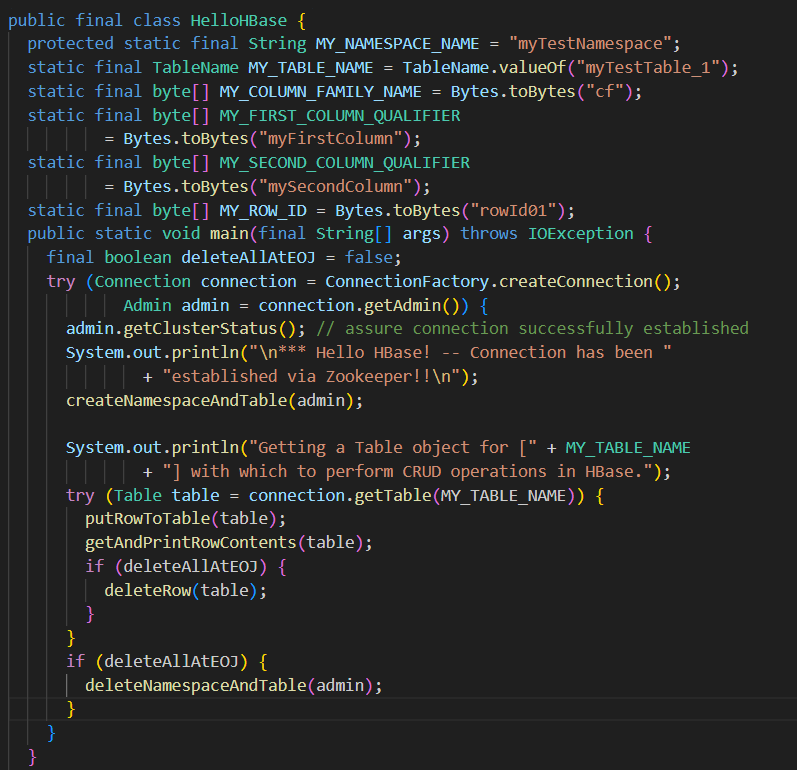
* Compile the java file.

3. java -cp $(hbase classpath):. HelloHBase

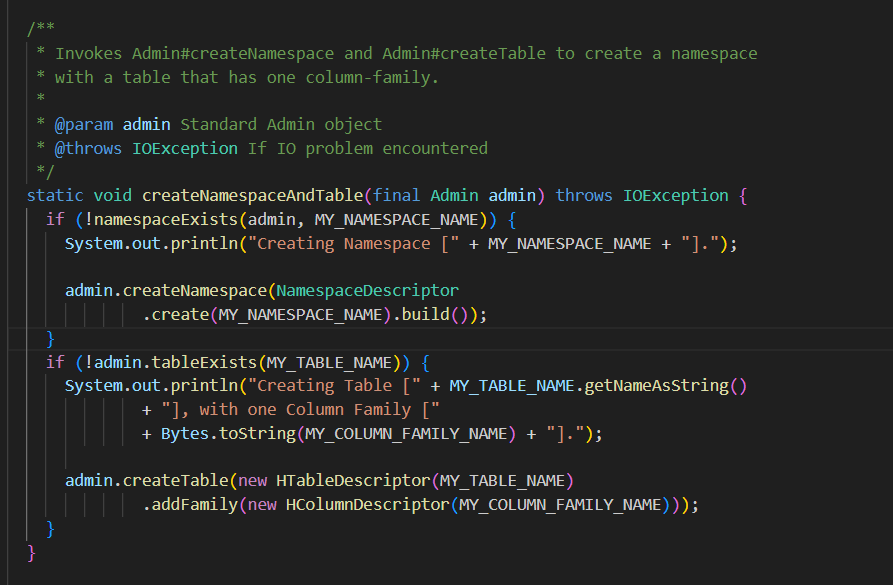
* Run the compile java class



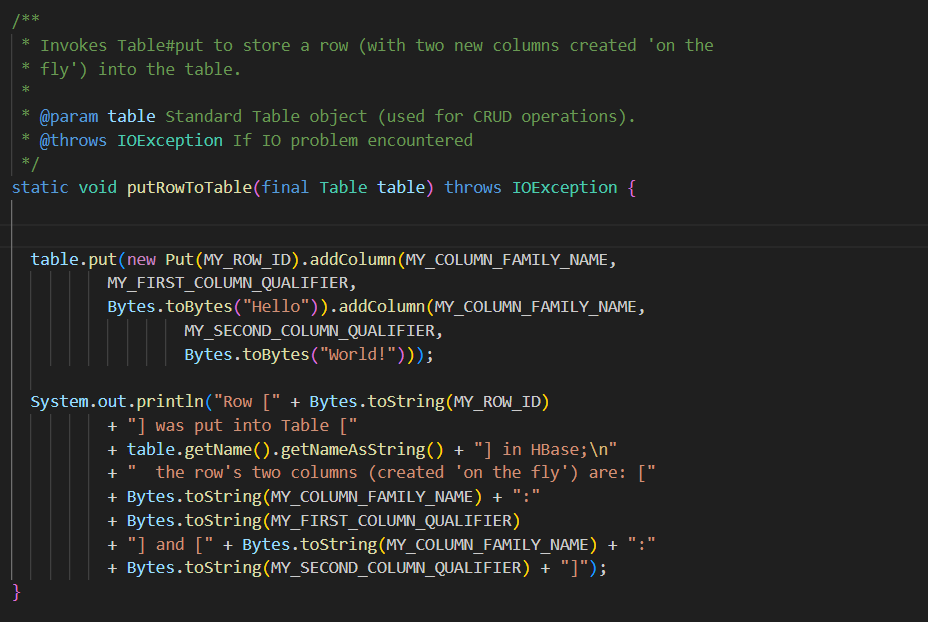
* Set of imports covers most of the common tasks needed for interacting with HBase in a Java application.



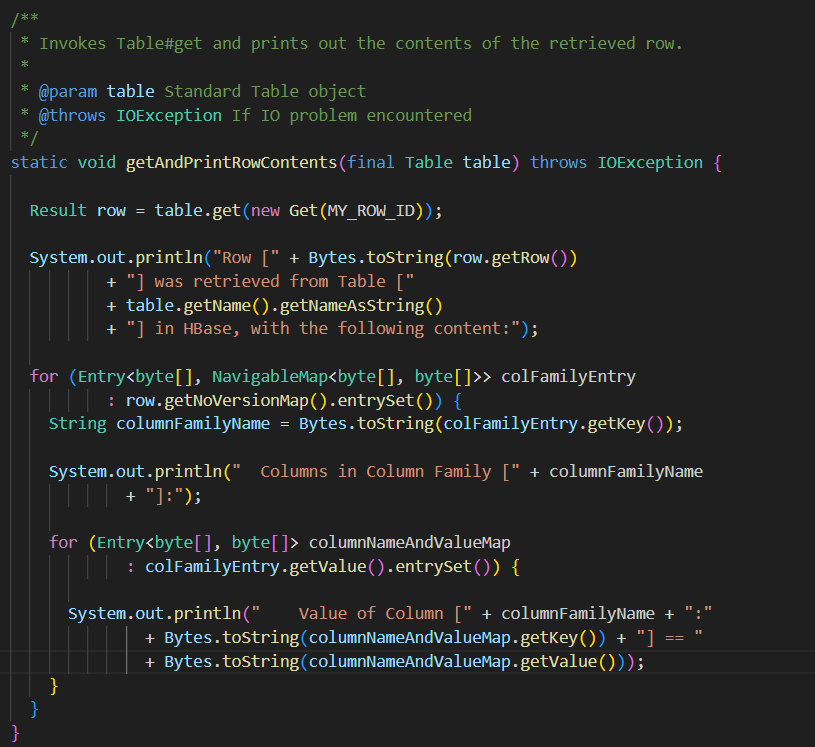
* ConnectionFactory #createConnection () automatically looks for hbase-site.xml (HBase configuration parameters) on the system's CLASSPATH, to enable creation of Connection to HBase via Zookeeper.
* Create a connection Using the connection, create an admin object, Using the admin object, perform Hbase functions
* This Java class demonstrates how to interact with HBase using the HBase client API. It connects to an HBase instance, creates a namespace and table, performs basic CRUD operations, and optionally cleans up resources.
* Define the namespace, table name, column family, column qualifiers, and row ID used in HBase operations.
* Creates a connection to HBase and retrieves an Admin object to perform administrative tasks.
* Creates Namespace and Table: Calls a method (not included in the snippet) to create the namespace and table if they don't already exist.
* Calls a method (not included) to put data into the table.
* Calls a method (not included) to get and print the row data.
* Conditionally deletes the row based on a flag.
* Optionally deletes the namespace and table if the flag is set.
* Uses try-with-resources to ensure that HBase connections and tables are properly closed after use.
* For the class to function correctly, methods to create the namespace and table, insert data, retrieve and print data, delete the row, and clean up resources need to be implemented.



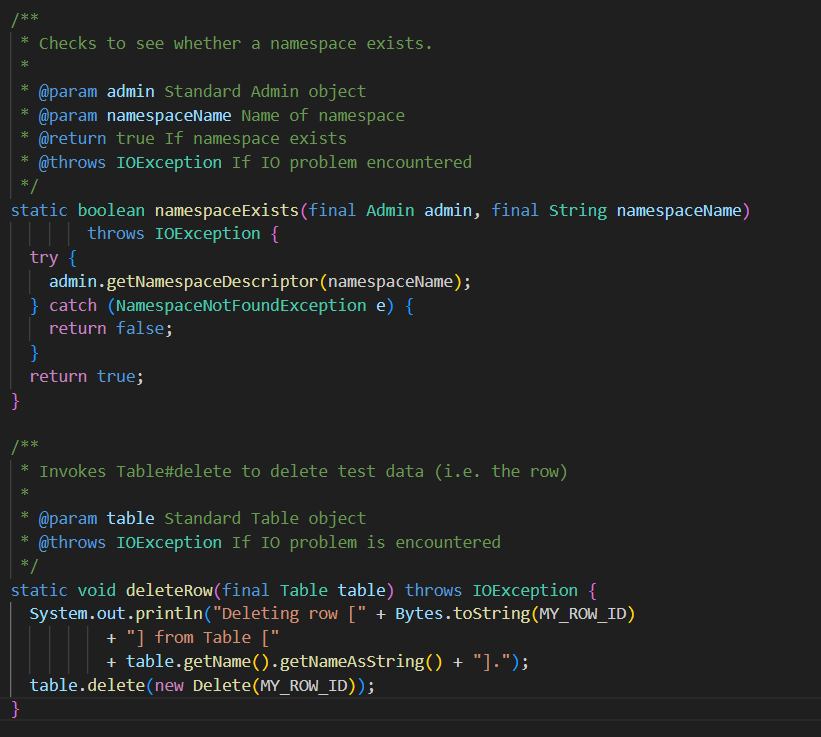
* createNamespaceAndTable method is designed to create a namespace and a table within HBase, provided they do not already exist.
* he method namespaceExists (which should be implemented separately) is used to check if the specified namespace already exists.
* If the namespace does not exist, it creates a new namespace using admin.createNamespace with a NamespaceDescriptor.
* Uses admin.tableExists(MY\_TABLE\_NAME) to check if the table already exists.
* If the table does not exist, it creates the table with one column family using admin.createTable. HTableDescriptor specifies the table, and HColumnDescriptor specifies the column family.
* namespaceExists(Admin admin, String namespaceName): This method should check whether the given namespace exists in the HBase instance.



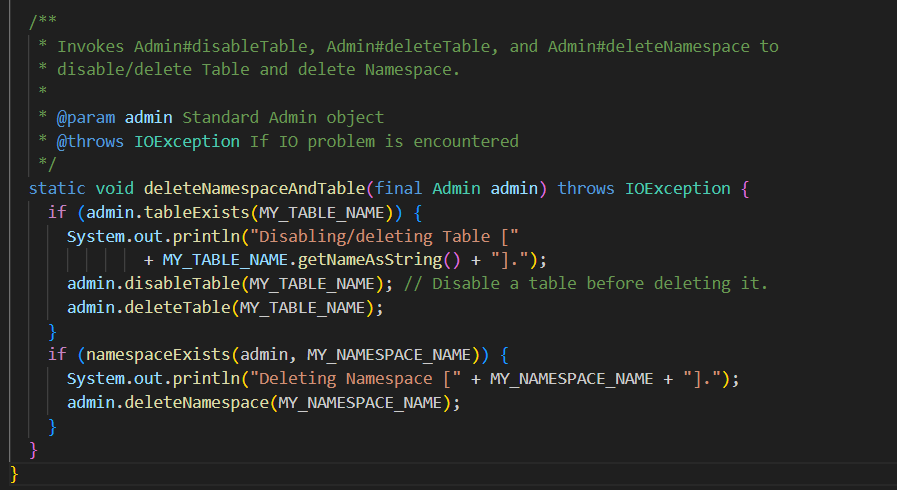
* putRowToTable method demonstrates how to insert a new row into an HBase table with specified column values
* Creates a Put object with the specified row ID (MY\_ROW\_ID). This object represents the row to be inserted or updated in the table
* Adds two columns to the Put object within the specified column family (MY\_COLUMN\_FAMILY\_NAME):
* First Column: With qualifier MY\_FIRST\_COLUMN\_QUALIFIER and value "Hello".
* Second Column: With qualifier MY\_SECOND\_COLUMN\_QUALIFIER and value "World!"
* Executes the put operation on the table, storing the row with the specified columns and values.
* Prints a message indicating that the row has been inserted into the table along with the column family and qualifiers



* getAndPrintRowContents method is designed to retrieve and display the contents of a row from an HBase table
* Creates a Get object for the specified row ID (MY\_ROW\_ID).
* Uses the Table#get method to retrieve the row. The result is stored in a Result object.
* Prints a message indicating that the row was retrieved from the table, including the row ID and table name.
* Iterates over the column families in the row using row.getNoVersionMap().
* For each column family, prints the name of the column family.
* Iterates over the columns and their values within each column family.
* Prints the column name (including column family and qualifier) and the corresponding value.



* The method attempts to retrieve the NamespaceDescriptor for the given namespaceName using admin.getNamespaceDescriptor(namespaceName).
* If the namespace does not exist, a NamespaceNotFoundException is thrown.
* If NamespaceNotFoundException is caught, the method returns false, indicating that the namespace does not exist.
* If no exception is thrown, it means the namespace exists, so the method returns true.
* Prints a message indicating that the row with ID MY\_ROW\_ID is being deleted from the specified table.
* Creates a Delete object for the row ID MY\_ROW\_ID.
* Uses the table.delete method to perform the deletion of the row.



* deleteNamespaceAndTable method is designed to clean up HBase resources by disabling and deleting a table and then deleting its associated namespace.
* Uses admin.tableExists(MY\_TABLE\_NAME) to check if the table exists.
* Before deleting a table, it must be disabled using admin.disableTable(MY\_TABLE\_NAME).
* Deletes the table using admin.deleteTable(MY\_TABLE\_NAME).
* Calls the previously defined namespaceExists method to verify if the namespace exists.
* Deletes the namespace using admin.deleteNamespace(MY\_NAMESPACE\_NAME).
* The HelloHBase class demonstrates basic HBase operations using Java.
* It connects to HBase, creates a namespace and table if they don't exist, performs CRUD operations by inserting a row and retrieving its contents, and optionally deletes the row and the namespace.
* The methods include creating and managing namespaces and tables, inserting and fetching data, and performing clean-up tasks.
* The example showcases typical interactions with HBase through its Admin and Table APIs, and handles exceptions for robust error management.