```
using System;
using System.Linq;
using System.Threading;
using Microsoft.Azure.Search;
using Microsoft.Azure.Search.Models;
using Microsoft.Extensions.Configuration;

namespace AzureSearchQuickstart
{
    class Program {
        // Demonstrates index delete, create, load, and query
        // Commented-out code is uncommented in later steps
        static void Main(string[] args)
        {
            IConfigurationBuilder builder = new
ConfigurationBuilder().AddJsonFile("appsettings.json");
            IConfigurationRoot configuration = builder.Build();

            SearchServiceClient serviceClient = CreateSearchServiceClient(configuration);

            string indexName = configuration["SearchIndexName"];

            Console.WriteLine("{0}", "Deleting index...\n");
            DeleteIndexIfExists(indexName, serviceClient);

            Console.WriteLine("{0}", "Creating index...\n");
            CreateIndex(indexName, serviceClient);
```

```
            // Uncomment next 3 lines in "2 - Load documents"
            // ISearchIndexClient indexClient = serviceClient.Indexes.GetClient(indexName);
            // Console.WriteLine("{0}", "Uploading documents...\n");
            // UploadDocuments(indexClient);

            // Uncomment next 2 lines in "3 - Search an index"
            // Console.WriteLine("{0}", "Searching index...\n");
            // RunQueries(indexClient);

            Console.WriteLine("{0}", "Complete.  Press any key to end application...\n");
            Console.ReadKey();
        }

        // Create the search service client
        private static SearchServiceClient CreateSearchServiceClient(IConfigurationRoot
configuration)
        {
            string searchServiceName = configuration["SearchServiceName"];
            string adminApiKey = configuration["SearchServiceAdminApiKey"];

            SearchServiceClient serviceClient = new SearchServiceClient(searchServiceName,
new SearchCredentials(adminApiKey));
            return serviceClient;
        }

        // Delete an existing index to reuse its name
```

```
        private static void DeleteIndexIfExists(string indexName, SearchServiceClient
serviceClient)
        {
            if (serviceClient.Indexes.Exists(indexName))
            {
                serviceClient.Indexes.Delete(indexName);
            }
        }

        // Create an index whose fields correspond to the properties of the Hotel class.
        // The Address property of Hotel will be modeled as a complex field.
        // The properties of the Address class in turn correspond to sub-fields of the
Address complex field.
        // The fields of the index are defined by calling the FieldBuilder.BuildForType()
method.
        private static void CreateIndex(string indexName, SearchServiceClient serviceClient)
        {
            var definition = new Index()
            {
                Name = indexName,
                Fields = FieldBuilder.BuildForType<Hotel>()
            };

            serviceClient.Indexes.Create(definition);
        }
    }
}
```