```
# Rewriting the Java code to a file again for user to download

collections_syntax = """

// HashMap Syntax and Usage

import java.util.HashMap;


public class CollectionsSyntax {


    // HashMap Declaration and Initialization

    HashMap<KeyType, ValueType> map = new HashMap<>(); // Create an empty HashMap

    HashMap<KeyType, ValueType> mapWithCapacity = new HashMap<>(initialCapacity); // Create a HashMap with initial capacity

    HashMap<KeyType, ValueType> mapWithLoadFactor = new HashMap<>(initialCapacity, loadFactor); // Create with load factor


    // Common HashMap Methods

    // Adding Key-Value Pairs

    map.put(key, value); // Adds a key-value pair, updates if key exists


    // Retrieving Values

    ValueType value = map.get(key); // Returns value for key or null if not found


    // Removing Key-Value Pairs

    map.remove(key); // Removes key-value pair for the key


    // Checking Existence

    boolean containsKey = map.containsKey(key); // Checks if key exists

    boolean containsValue = map.containsValue(value); // Checks if value exists
```

```
// Size of the Map
int size = map.size(); // Returns number of key-value pairs

// Clearing the Map
map.clear(); // Removes all key-value pairs

// Iterating Over the Map
for (KeyType key : map.keySet()) {
    ValueType value = map.get(key); // Iterate over keys
}

for (ValueType value : map.values()) {
    // Do something with value // Iterate over values
}

for (Map.Entry<KeyType, ValueType> entry : map.entrySet()) {
    KeyType key = entry.getKey(); // Iterate over key-value pairs
    ValueType value = entry.getValue();
}

// HashSet Syntax and Usage
import java.util.HashSet;

// HashSet Declaration and Initialization
HashSet<Type> set = new HashSet<>(); // Create an empty HashSet
HashSet<Type> setWithCapacity = new HashSet<>(initialCapacity); // Create a HashSet with initial capacity
HashSet<Type> setWithLoadFactor = new HashSet<>(initialCapacity, loadFactor); // Create with load factor
```

```
// Common HashSet Methods
// Adding Elements
set.add(element); // Adds an element to the set

// Removing Elements
set.remove(element); // Removes specified element

// Checking Existence
boolean contains = set.contains(element); // Checks if set contains element

// Size of the Set
int size = set.size(); // Returns number of elements

// Clearing the Set
set.clear(); // Removes all elements

// Iterating Over the Set
for (Type element : set) {
    // Do something with element // Iterate over elements
}

// TreeMap Syntax and Usage
import java.util.TreeMap;

// TreeMap Declaration and Initialization
TreeMap<KeyType, ValueType> treeMap = new TreeMap<>(); // Create an empty TreeMap
```

```
  TreeMap<KeyType, ValueType> treeMapWithCapacity = new
TreeMap<>(initialCapacity); // Create with initial capacity

  // Common TreeMap Methods
  // Adding Key-Value Pairs
  treeMap.put(key, value); // Adds key-value pair, updates if key exists

  // Retrieving Values
  ValueType value = treeMap.get(key); // Returns value for key or null if not found

  // Removing Key-Value Pairs
  treeMap.remove(key); // Removes key-value pair for the key

  // Checking Existence
  boolean containsKey = treeMap.containsKey(key); // Checks if key exists
  boolean containsValue = treeMap.containsValue(value); // Checks if value exists

  // Size of the TreeMap
  int size = treeMap.size(); // Returns number of key-value pairs

  // Clearing the TreeMap
  treeMap.clear(); // Removes all key-value pairs

  // Iterating Over the TreeMap
  for (Map.Entry<KeyType, ValueType> entry : treeMap.entrySet()) {
    KeyType key = entry.getKey(); // Iterate over key-value pairs
    ValueType value = entry.getValue();
  }
```

```
}
"""

# Save the collections syntax to a Java file
collections_file_path = "/mnt/data/CollectionsSyntax.java"
with open(collections_file_path, "w") as file:
    file.write(collections_syntax)

collections_file_path
```