Collections Concepts (Important for Automation)

- ## Interfaces:

List
Set
Map
Queue

- ## Classes:

ArrayList, LinkedList
HashSet, LinkedHashSet, TreeSet
HashMap, LinkedHashMap, TreeMap
PriorityQueue, Deque

- ## Utility Classes:

Collections (sorting, searching, synchronizing)
Arrays (for array-to-list conversions, sorting)

Common Collections Interview Questions (with Automation Context)

1. What is the difference between List and Set?

- List: Allows duplicates, maintains insertion order
- Set: No duplicates, unordered (unless LinkedHashSet or TreeSet)
- Used In Automation: Store test data (List), remove duplicates (Set)

2. Why do we use ArrayList in Selenium frameworks?

- Dynamic sizing
- Fast access using index
- Common for storing List<WebElement> from findElements()

3. How do you store key-value data in automation tests?

- Use HashMap<String, String>
- Example: Map<String, String> testData = new HashMap<>();
- Store config values, test data, API headers, etc.

Common Collections Interview Questions (with Automation Context)

4. How do you handle duplicate test data in your framework?

- Use Set<String> to store data
- Automatically removes duplicates (e.g., dropdown values)

5. Where have you used Maps in your automation framework?

- Store element locators
- Store input data from Excel/JSON
- Store expected vs actual results

6. What's a real use case of TreeMap or TreeSet in automation?

- TreeMap: When sorted keys are needed (e.g., module-wise test data)
- TreeSet: To store sorted, unique test identifiers or names

Common Collections Interview Questions (with Automation Context)

7. How do you remove duplicates from a List?

- List<String> unique = new ArrayList<>(new HashSet<>(originalList));

8. How do you convert:

- List to Set: Set<T> set = new HashSet<>(list);
- Set to List: List<T> list = new ArrayList<>(set);
- Map keys to List: new ArrayList<>(map.keySet());

9. How do you ensure thread safety in collections?

- use Collections.synchronizedList(), synchronizedMap()
- Apply in parallel test executions

Collections in Automation Testing - Mostly asked Use Cases in Selenium

1. How do you store WebElements returned by findElements()?

- Use: List<WebElement>

driver.findElements(By.xpath(...)) returns a list of matching elements.

You can loop through this list to validate dropdown items, menu links, etc.

2. How do you manage test data in a data-driven framework?

- Map<String, String> for a single row of test data (column name → value)
- List<Map<String, String>> for multiple rows of test data
  - Works well when reading data from Excel, JSON, or CSV
  - Allows easy mapping of test case ID → corresponding inputs and expected values

Collections in Automation Testing - Mostly asked Use Cases in Selenium

3. How do you store test steps or reusable locators?

Use: HashMap<String, By> or Map<String, String>

- Map helps manage element locators by logical names
- Reusable across tests → improves maintainability of Page Object Models

4. Where do you use Set in Selenium frameworks?

Useful for storing unique values like:

- All window handles (driver.getWindowHandles())
- Unique dropdown items to avoid duplicates
- Prevents duplication without needing manual checks

Java Collections in API Testing – Key Interview Questions

1. How do you pass headers in an API request using Collections?

- Use Map<String, String> to store headers
- Pass it to .headers() in RestAssured

Example:

```
Map<String, String> headers = new HashMap<>();
headers.put("Authorization", "Bearer token");
headers.put("Content-Type", "application/json");
given().headers(headers);
```

2. How do you build dynamic JSON payloads using Collections?

- Use Map<String, Object> for flexible JSON body
- Add keys and values dynamically
- Works well with POST and PUT requests

Example:

```
Map<String, Object> body = new HashMap<>();
body.put("name", "John");
body.put("age", 30);
given().body(body).post("/users");
```

Java Collections in API Testing – Key Interview Questions

3. How do you represent a JSON array of objects in Java?

- Use List<Map<String, Object>>
- Each Map is one object in the array
- Useful for batch POST or validation

Example:

```
List<Map<String, Object>> payload = new ArrayList<>();
Map<String, Object> item1 = new HashMap<>();
item1.put("id", 1); item1.put("name", "A");

Map<String, Object> item2 = new HashMap<>();
item2.put("id", 2); item2.put("name", "B");

payload.add(item1);
payload.add(item2);
```

Java Collections in API Testing – Key Interview Questions

4. How do you extract multiple values from an API response?

- Use List<String> to collect values using jsonPath()

Example:

```
List<String> ids =
response.jsonPath().getList("users.id");
```

5. How do you validate uniqueness in API responses?

- Use Set<String> to store values
- Compare size of Set and List to ensure no duplicates

Example:

```
List<String> emails =
response.jsonPath().getList("data.email");
Set<String> uniqueEmails = new HashSet<>
(emails);

assert emails.size() == uniqueEmails.size();
```

Java Collections in API Testing – Key Interview Questions

6. How do you manage multiple responses in a test flow?

- Use Map<String, Response>
- Helps store and reuse responses from different endpoints

Example:

```
Map<String, Response> apiResponses = new HashMap<>();
apiResponses.put("createUser", res1);
apiResponses.put("getUser", res2);
```

7. Why are Collections important in API automation frameworks?

- Enable dynamic request generation
- Help in building data-driven tests
- Useful for parsing and validating complex JSON responses
- Support clean and reusable code

High-Impact Interview Questions

1. What happens if two keys have the same hashcode in a HashMap?

- Both keys go to the same bucket.
- HashMap uses equals() to resolve the key collision.
- If keys are equal, value is updated.
- If keys are different, a linked list or tree stores them.

2. Why is Map not a part of the Collection interface?

- Collection represents a group of elements (single values).
- Map represents key-value pairs, which is a different structure.
- Therefore, Map doesn't extend Collection.

High-Impact Interview Questions

3. What is the difference between equals() and hashCode()?

- equals()
  - Used to compare the actual content (or state) of two objects.

    Example:

```
String a = new String("test");
String b = new String("test");
System.out.println(a.equals(b)); // true
```

- hashCode()

  Returns an integer value representing the object's memory location or calculated hash.

  Example:

```
System.out.println(a.hashCode()); // e.g., 3556498
System.out.println(b.hashCode()); // same hashCode as `a`
```