Detailed Notes on *Collections in Java*

Introduction to Collections

- A **collection** is a group of objects (called elements) stored together in a single container.
- **Initial History**:
  - Originally known as `Java.util package` or `Collection API`.
  - Officially introduced as a **collection framework** in Java 1.2.
  - Designed by *Joshua Bloch*.
- Enhanced in Java 1.5 with the introduction of **Generics**.
- Collections are containers to hold multiple objects and are often referred to as **container objects**.

Difference Between Arrays and Collections

Aspect	Arrays	Collections
Size	Fixed	Growable
Memory Efficiency	Less efficient	Highly efficient
Performance	Better	Comparatively slower
Data Type Support	Homogeneous only	Both homogeneous and heterogeneous
Underlying Structure	None	Standardized data structures
Primitives Support	Supports primitives and objects	Supports only objects

---

Types of Objects Stored in a Collection

1. **Homogeneous Objects**: Objects of the same type (e.g., multiple `Student` objects).
2. **Heterogeneous Objects**: Objects of different types (e.g., `Student` and `Employee` objects).
3. **Duplicate and Unique Objects**:
   - **Duplicate**: Objects with the same value (e.g., two `Person` objects named "John").
   - **Unique**: Objects with distinct values.

Advantages of Using Collections

1. **Ease of Development**: Reduces coding effort and time.
2. **Growable Nature**: Container size adjusts dynamically.
3. **High Performance**: Efficient data structures and algorithms.
4. **Code Reusability**: Provides reusable interfaces and classes.

Java Collections Framework

- A framework consisting of **interfaces** and **implementation classes**.
- Handles objects as a single entity.

- Present in the `java.util` package.
- Provides operations such as storing, retrieving, and updating objects.

Core Interfaces and Implementation Classes

- **Key Interfaces**: `List`, `Set`, `Queue`, `Map`.
- **Key Classes**: `ArrayList`, `LinkedList`, `HashSet`, `TreeSet`, `PriorityQueue`, `HashMap`, `TreeMap`.

List Interface

- **Definition**: Ordered collection that maintains elements in sequential order and allows duplicates.
- **Key Features**:
  - Uses **index-based structure**.
  - Supports duplicate elements.
  - Maintains **insertion order**.
  - Allows multiple `null` elements.
  - Provides `ListIterator` for bidirectional traversal.
- **Implementation Classes**: `ArrayList`, `LinkedList`, `Vector`, `Stack`.

Set Interface

- **Definition**: Unordered collection of **unique elements**.
- **Key Features**:
  - Does not allow duplicates.
  - No **index-based access**.
  - Most implementations allow one `null` element.
  - Uses **map-based structures**.
- **Implementation Classes**: `HashSet`, `LinkedHashSet`, `TreeSet`.

Key Classes

1. **ArrayList**:
   - Dynamic, resizable array.
   - Supports duplicate and `null` elements.
   - **Not synchronized** (not thread-safe).
   - Provides **random access**.
2. **LinkedList**:
   - Implements a **doubly linked list** structure.
   - Suitable for frequent insertion/removal in the middle.
   - Does not support **random access**.
3. **HashSet**:
   - Unordered collection of unique elements.
   - Backed by a **hash table**.
   - Allows one `null` value.
4. **TreeSet**:
   - Ordered collection of unique elements.
   - Uses a **red-black tree** structure.

- Maintains **ascending order**.

5. **Vector**:
   - Synchronized and thread-safe.
   - **Legacy class** introduced in JDK 1.0.

Queue Interface

- **Definition**: First-In-First-Out (FIFO) structure.
- **Key Features**:
  - Elements are added at the tail and removed from the head.
  - Does not allow `null` elements.
- **Implementation Classes**: `LinkedList`, `PriorityQueue`, `ArrayDeque`.

Map Interface

- **Definition**: Key-value pair collection where keys are unique.
- **Key Features**:
  - Duplicate keys are not allowed; duplicate values are permitted.
  - No **index-based access**.
  - **Subtypes**: `HashMap`, `LinkedHashMap`, `TreeMap`.

Comparable vs Comparator

Aspect	Comparable	Comparator
Sorting	Single criteria	Multiple criteria
Method Used	`compareTo()`	`compare()`
Influence on Class	Modifies original class	External class can define logic
Package	`java.lang`	`java.util`

When to Use

- **Comparable**: For **natural sorting** (e.g., sorting `Employee` by ID).
- **Comparator**: For **custom sorting** (e.g., sorting `Employee` by name and age).

Summary of Features

Feature/Interface	Allows Duplicates	Maintains Order	Thread-Safe	Null Elements
ArrayList	Yes	Yes	No	Yes
LinkedList	Yes	Yes	No	Yes
HashSet	No	No	No	One
TreeSet	No	Yes (Sorted)	No	No
Vector	Yes	Yes	Yes	Yes

Let me know if you'd like a tailored MCQ question bank or further detailed comparisons!