1.how can we access the inner class members?

```
package com.aspiresys;
class Outer{
	static class Inner{
		int num =9;
	}
}
class Child extends Outer{
	void display() {
		Outer.Inner inner = new Outer.Inner();
		System.out.println("Hi");
		System.out.println(inner.num);
	}
}
class Main {
  public static void main(String[] args) {
    Child child = new Child();
    child.display();
  }
}
```

2. can we create a class as private?

In java we cannot create class as private because it would not be accessible to any other class, not even classes within the same package. But we can declare an inner class as private, meaning it can only be accessed within the outer class.

3.How many child class objects can be created for super class?

There is no fixed number of child class objects that can be created for a superclass. The limit is determined by available system resources (such as memory) and practical design considerations.

4.Why Map not included in collection?

Map is designed to handle key-value pair associations, which is a different kind of structure from the element-based collections (List, Set, Queue)

Collection deals with individual items but map deals with pair (key and value)

5.Is dictionary class available in java?

In Java, there is no Dictionary type in java, but Java provides similar functionality by Map, the Map interface (and its implementations like HashMap, TreeMap, and LinkedHashMap) serves the same purpose and is the recommended approach for storing key-value pairs.

6.Difference between collections class and Collection Interface?

| Collections class | Collection Interface |
|---|---|
| The `Collections` class is a utility class that provides static methods to operate on or return collections. It doesn't define the structure of collections but offers various utility functions. | The `Collection` interface is the root of the collection hierarchy in Java. It defines the fundamental operations that all collection classes must support. |
| `Static methods that provide utility operations on collections. These methods don't alter the structure of the collection but perform actions like sorting, reversing,etc...`<br>`-sort(List<T> list)`<br>`- reverse(List<T> list)`<br>`- shuffle(List<T> list)` | Instance methods that are defined to be implemented by any class that implements the `Collection` interface include:<br>`- add(E e)`<br>`- remove(Object o)`<br>`- clear()`<br>`- contains(Object o)`<br>`- iterator()`<br>`- size()`<br>`- isEmpty()` |

7.Implementation of proirity queue?

```
import java.util.PriorityQueue;

public class PriorityQueueExample {
```

```
   public static void main(String[] args) {
      // Creating a priority queue
      PriorityQueue<Integer> priorityqueue = new PriorityQueue<>();
      // Inserting elements into the priority queue
      priorityqueue.add(10);
      priorityqueue.add(30);
      priorityqueue.add(20);
      priorityqueue.add(40);
      System.out.println("Priority Queue: " + priorityqueue);
      // Deleting elements (poll removes the element with the highest priority, i.e., the
smallest element)
      System.out.println("Removed: " + priorityqueue.poll());  // Should remove 10 (smallest)
      System.out.println("Removed: " + priorityqueue.poll());  // Should remove 20
      // Display the current Priority Queue
      System.out.println("Priority Queue after removals: " + priorityqueue);
      // Peek: Returns the element with the highest priority (smallest element) without
removing it
      System.out.println("Element at the top (peek): " + priorityqueue.peek());
   }
}
```

8.Implementation of Deque?

```
import java.util.*;
public class DequeExample {
   public static void main(String[] args) {
      // Create a Deque using ArrayDeque
      Deque<Integer> deque = new ArrayDeque<>();
      // Adding elements to the deque
      deque.addFirst(10); // Add element at the front
      deque.addLast(20);  // Add element at the rear
```

```
        deque.addFirst(5);  // Add element at the front
        deque.addLast(30);  // Add element at the rear
        // Display the deque
        System.out.println("Deque after additions: " + deque);
        // Removing elements from the deque
 System.out.println("Removed from front: " +deque.removeFirst());  // Remove element from the front
System.out.println("Removed from rear: " + deque.removeLast());    // Remove element from the rear
        // Display the deque after removals
        System.out.println("Deque after removals: " + deque);
        // Peek elements from both ends without removing
     System.out.println("Peek at front: " + deque.peekFirst());  // Peek element from the front
      System.out.println("Peek at rear: " + deque.peekLast());    // Peek element from the rear
    }
}
```

9.What is Thread safe?

Thread safety in Java refers to the ability of a class or object to function properly when multiple threads access it simultaneously.

10.Difference between HashMap, HashSet, TreeSet,TreeMap?

| Feature | HashSet | TreeSet | HashMap | TreeMap |
|---|---|---|---|---|
| Implements | `Set<E>` | `Set<E>` | `Map<K, V>` | `Map<K, V>` |
| Allows Duplicates | No (elements are unique) | No (elements are unique) | No (keys are unique) | No (keys are unique) |
| Order | No specific order | Sorted (natural/comparator order) | No specific order | Sorted by keys (natural/comparator order) |
| Null Elements Allowed | One `null` element | No `null` elements | One `null` key, many `null` values | No `null` keys, but allows `null` values |
| Underlying Data Structure | Hash table | Red-Black Tree | Hash table | Red-Black Tree |

| | | | | |
|---|---|---|---|---|
| Use Case | Unique elements, fast lookups | Unique elements with sorted order | Key-value mapping, fast lookups | Key-value mapping with sorted keys |

11. what are the in-built interfaces present in java?

Comparable, serializable, Runnable, callable, Iterable, collection, comparator, cloneable, List, Set, Map, Queue, Deque, Stream,Iterator

12.what is throw and throws:

Throw:

The throw keyword is used to explicitly throw an exception from a method or any block of code.

The throw keyword is used to create a custom error. The throw keyword is mainly used to throw custom exceptions.

Throws:

The throws keyword is used in the method to indicate that the method might throw one or more exceptions

This informs the caller of the method that they need to handle these exceptions, either by using a try-catch block or by declaring the exceptions in their own method.

13.when to use throw an throws:

* When there is an checked exception occured.

*throws is used in a method so that whenever there is an exception, the throws will notify the user that there has been an exception occurred and you need to handle the exception.

14.why throws and throw:

whenever you are getting some information from the user ,some times user may enter the details that does not match our coding need ,in such case the throw and throws will handle the exception by talking the value has a null value ,but the exception maybe not handled properly.

15.why interface used for base class?

- Interfaces are used instead of classes to define expected behavior without enforcing any specific implementation.
- They allow for multiple inheritance, making code more flexible and decoupled.
- They promote separation of concerns, ensuring that classes focus on implementing behavior, while interfaces specify the required actions.
- Classes define both state and behavior, which can create tight coupling and reduce flexibility. Interfaces, on the other hand, only define behavior, leaving the details to the implementing classes.
- Using interfaces makes your system more extensible and maintainable, as they allow new behavior to be added without modifying existing code.