13. Java-related

Some important stuff:

Overloading vs Overriding

Overloading is used to describe when two methods have the same name but differ in the type or number of arguments.

Overriding occurs when a method shares the same name and function signature as another method in its super class.

Collections Framework

ArrayList: An ArrayList is a dynamically resizing array, which grows as you insert elements.

Vector: A vector is very similar to an ArrayList, except that it is synchronized.

LinkedList: Java's built-in LinkedList class. Rarely comes up in an interview, but it's useful to study because it demonstrates some of the syntax for an iterator.

HashMap

Questions

Question 1: Private Constructor. In terms of inheritance, what is the effect of keeping a constructor private?

Declaring a constructor private on class A means that you can only access the constructor if you could also access its other private methods. A's inner classes can access these. Additionally, if A is an inner class of Q, then Q's other inner classes also have access to them.

This has direct implications for inheritance since a subclass calls its parent's constructor. Class A can be inherited, but only by its own or its parent's inner classes.

Question 2: Return from Finally. In Java, does the finally block get executed if we insert a return statement inside the try block of a try-catch-finally?

Yes , it will get executed. When we attempt to exit the try block (via a return statement, a continue statement, a break statement, or any exception), the finally block will still be executed.

There are some cases in which the finally block will not get executed:

- If the VM exits during the try/catch block execution
- If the thread executing the try/catch block gets killed

Question 3: What is the difference between final, finally and finalize?

Final statement has a different meaning depending on its context:

- When applied to a variable (primitive): The value of the variable cannot change
- When applied to a variable (reference): The reference variable cannot point to any other object on the heap
- When applied to a method: The method cannot be overridden.
- When applied to a class: The class cannot be subclassed.

Finally is a block that can be used at the end of a try-catch block that will always get executed after either the try/catch block finishes. The finally block is often used to write the clean-up code. The catch block is fully executed (including the function call in the return statement), then the finally block, and then the function actually returns.

Finalize() is a method called by the garbage collector once it determines that no more references exist. It is called just before actually destroying the object.

Question 4: Difference between Java Generics and C++ templates?

- C++ templates can use primitive types, like int. Java cannot and must instead use the wrapper class Integer.
- In Java, you can restrict the template's type parameters to be of a certain type. In C++, the type parameter can be instantiated. Java does not support this.
- In Java, the type parameter (i.e., the Foo in MyClass) cannot be used for static methods and variables, since these would be shared between MyClass and MyClass.
- In Java, all instances of MyClass, regardless of their type parameters, are the same type. The type parameters are erased at runtime.

Question 5: Differences between TreeMap, HashMap and LinkedHashMap

All offer a key -> value map and a way to iterate through the keys. The most important distinction between these classes is the time guarantees and the ordering of the keys.

- HashMap offers O(1) lookup and insertion. If you iterate through the keys, the ordering of the keys is essentially arbitrary: It is implemented by an array of linked lists.
- TreeMap offers O(log N) lookup and insertion. Keys are ordered, so you need to iterate through the keys in sorted order. Keys must implement the Comparable interface. TreeMap is implemented by a Red-Black Tree.
- LinkedHashMap offers O(1) lookup and insertion. Keys are ordered by their insertion order. It is implemented by doubly-linked buckets.

Imagine passing an empty TreeMap, HashMap and LinkedHashMap into the following function:

```
void insertAndPrint(AbstractMap<Integer, String> map) {
  int[] array = {1, -1, 0};
  for (int x : array) {
    map.put(x, Integer.toString(x));
  }
  for (int k : map.keySet()) {
```

```
      System.out.print(k + ", ");
    }
  }
```

HashMap: (any ordering)
LinkedHashMap: {1, -1, 0}
TreeMap: {-1, 0, 1}

Generally, unless there is a reason not to, you would use HashMap. It is typically faster and requires less overhead. If you need to get the keys back in insertion order, then use LinkedHashMap. If you need to get the keys back in their true/natural order, then use TreeMap.

Question 6: What is object reflection and why it is useful

Object reflection is a feature in Java that provides a way to get reflective information about Java classes and objects and perform operations such as:

1. Getting information about the methods and fields present inside the class at runtime.
2. Creating a new instance of a class.
3. Getting and setting the object fields directly by getting field reference, regardless of the access modifier

Object reflection is useful because:

1. It can help you observe or manipulate the runtime behaviour of applications.
2. It can help you debug or test programs as you have direct access to methods, constructors and fields.
3. You can call methods by name when you don't know the methods in advance.