summarize missing points:

4. \*\*Template Types in Java\*\* including `class`, `interface`, and `abstract class`, along with `enum` and `annotations(@interface)`

`abstract classes` can contain abstract methods without implementation bodies and that inheriting from them requires overriding.

`abstract classes` allow only single inheritance, whereas `interfaces` permit multiple implementations

`class MyClass implmeent interface1, 2, 3 extends AbstractClass`

6. five key areas in JVM memory architecture: stack, heap, program counter, method area, and native method stack.

`static` elements, including class templates and source code, are stored in the method area, while object instances reside in the heap.

static blocks are executed once during program loading and are used for bootstrap logic.

`template -> access elements from template`

Each object accesses class-level elements from the class template stored in the Method Area.

`JVM -> Java Virtual Machine -> type of virtual machine(Process Virtual Machine)`

7. \*\*Immutability and the Final Keyword.\*\*

`final -> field, method, class`

for method, `final` means prevents overriding.

for class, `final` means prevents inheritance. cannot be subclassed.

for fields, `final` means the reference cannot be reassigned, distinguishing it from an immutable object.

(final→reference cannot be reassigned; immutable→object's content can not be changed.)

between eager and lazy loading in singleton patterns, noting `final` cannot be used with lazy loading because the instance is assigned later.

using getter methods to control access and maintain immutability; deep copies for reference data types in getters to prevent unintended modifications

\*\*`finalize`\*\*: A method called by the garbage collector before an object is destroyed, typically used (though not recommended) to release resources.

\*\*`finally`\*\*: A block of code that always executes after a `try` block, regardless of whether an exception occurs, usually for cleanup tasks.

8. immutable class means \*\*instances cannot be modified after they are created\*\*. Once you create an object of an immutable class, its state (its fields) cannot change.

The class is declared `final` (so it cannot be subclassed).

All fields are `private` and `final`.

No setter methods are provided.

All fields are initialized via the constructor.

If a field is mutable (like a `Date` or a `List`), the class makes a \*\*defensive copy\*\* when setting or returning it.

13. `8 primitive data type`:byte < short < int < long，float < double，char，boolean

（1 byte < 2 byte < 4 byte < 8 byte，4 byte < 8 byte，2 byte，1 byte/1 bit）

`auto boxing`Automatic conversion from a \*\*primitive type\*\* to its \*\*wrapper class\*\*.

`auto unboxing`Automatic conversion from a \*\*wrapper class\*\* back to its \*\*primitive type\*\*.

Useful when working with \*\*collections\*\* or \*\*arithmetic operations\*\*.

15. map implementation?