Interview Questions

This test should take you between 30 and 60 minutes, and you should aim to write between 20 and 100 words per answer. If you're not sure about an answer, just write what you think, however don't wildly guess. It is better to say "I don't know" rather than to have a completely wrong answer.

Good Luck!

1. Is Java pass by reference or pass by value? What especially should you be aware of?

   Java uses "**pass by value**" approach when passing values to a function. With that said, there is one caveat we should pay attention to.

   If a variable passed into the function as an argument having **primitive type** like "int", the actual value which is in the stack will be copied and then passed into the function, which means the original variable's value won't be changed by the change made in the function to that variable.

   If the variable passed into the function **is a reference** (in stack as well) pointing to a **mutable** object in the heap, the reference per se will be copied and then passed into the function, which means there will be two references in the stack **pointing to exactly the same object** in the heap. In this case, any changes in the function which update the state of the object, say, changing the value of an attribute via setter, will be visible outside the function, this is called "side effect".

2. What is method overriding and method overloading? How are they different?

   Overriding means the sub-class changes the behaviour of the method having the same prototype (signature) inherited from the base class. It is used in **dynamic binding** to embody the polymorphism. For example, the abstract class HttpServlet's doGet() method can be overrided by sub-classes.

   Overloading means a class has multiple functions having same method name but different number of arguments, data type of arguments. For example, a class might have multiple overloaded constructors.

3. Can you provide an example of the 'Chain of Responsibility' design pattern that you have used recently?

   Recently I've used 'Chain of Responsibility' pattern to implement the Quick Reference Guide PDF report service. AEM server sends a JSON payload representing concrete product (for example, a refrigerator) data, which has multiple sections, by HTTP POST, and the service parses the JSON payload via iText and then generates the PDF file. Since the product data contain multiple sections, I split the logic into several smaller classes called "xxxSectionPopulator" which are responsible for populating different sections respectively and then made the service entry class link those populators via "setSuccessor()" method so as to generate the final PDF file. By leveraging the chain of responsibility in this scenario, the content in the PDF file can be adjusted only through simply changing the chain order, or adding new populator into the chain. Flexibility has been achieved.

4. Can you give examples from when you would make use of the Observer design pattern?

   Example1: In the message queue, **publish/subscribe mode** uses the Observer design pattern. Message broker is the observable, whereas the subscriber is the observer. Once there's new message in a specific topic, the message broker sends the message to the subscriber.

   Example 2: The **EventListener** observes the event triggered on a button, say, submit button in the shopping cart. Once the onClick event happens, the eventListener will be notified and then do something business logic like validating the stock level of the products placed into the cart.

5. Explain how you apply the principles of DRY in your daily work.

   DRY – Don't Repeat Yourself. In my daily work, I carefully review the code written by me or others, make sure there is no duplicated code, if so, I'll refactor it via well-known design patterns like Template, Strategy, Vistor etc... to extract common method, and also modify the unit test the accommodate the changes.

   In addtion, in the continueous integration environment, I prefer setting up tools like SonarQube to run the report of code quality regarding duplicated code, and also the SonarQube plugin in Eclipse to watch my code.

6. Describe how you go about writing unit tests.
   Unit test is the lowest level in the testing pyramid, which contain unit test, integration test and end-to-end test. Unit test is responsible for testing each method in a class. Normally I would like to try Test Driven Development, writing unit test for the functionality a method needs to achieve, and then write the method to make all the test cases pass (see the green bar). In principle, I prefer using unit test to write all possible cases including edge cases. Mockito, PowerMock are the tools I used to mock some objects and then stub some methods in the object which I don't want to test.

7. What is most frustrating to you about poorly or inconsistently formatted code?
   Poorly or inconsistently formatted code introduce bad readability and smell when I do the reverse engineering to figure out the actual purpose of the code. It violates the "Boy Scout" rule, and then reduces the productivity of other team members, because others need to spend more time and effort reading the bad code. Imagine there're no indentions in nested if blocks and for loops. As a senior developer or tech lead, I designed the unique Eclipse code formatter preference file for the team.

8. Why is important to understand an object's lifecycle? What are the implications of storing user sensitive information in a web application scoped context, for example?

   Clearly understanding an Object's lifecycle will definitely help us to design our algorithms to make better use of memory, reduce the memory footprint as well as avoid memory leak. For example, in Java, understanding Garbage Collection is the crucial part of eliminating memory leak.

   Implications of storing user sensitive information in a web application scoped context:

   - Storing clear text user info in browsers' cookies. If it is being passed outside of ssl or https then man in middle can clearly read. What security is being used.
   - If I can see the clear text user info in a core dump file or heap dump of JVM when the system crashes.

9. What is the difference between .equals() and ==?

   a. equals() is a method, whereas == is an operator;

   b. == is used for a reference (address) comparison, while equals() method for content comparison based on the implementations.

   c. By default, if a class directly extends Object class without overriding equals() method by comparing content, then equals() method and == will return the same value simply because the source code of equals() method in Object class:

```
public boolean equals(Object obj) {

    return (this == obj);

}
```

10. Explain the following line of code
    public static void main(final String... args)

    This is the main method of a class, which accepts zero or more arguments. If the java file containing the main() method is compiled to class file (bytecode), and the class file is executed by "java" command, this method will be invoked by the main JVM thread. This method is static, which means it belongs to the class and is shared by all objects of the class. "void" means this method returns nothing.

11. What's good about your favourite IDE?
    My two favourite IDEs are Eclipse and Microsoft Visual Studio Code. Eclipse is based on Equinox OSGi framework, and it is a "microkernel" system per se. It provides many useful plugins and beautiful themes which can boost programmers' productivities. Also, Eclipse is open source, which help me to learn how to write clean code and better idea. Visual Studio Code is another example of microkernel system which provides many plugins. It is lightweight comparing with Microsoft Visual Studio. I usually use Visual Studio Code to write Python, C/C++ as well as Javascript code.

12. What is your build tool of choice? Maven, grade, ant, something else? Why?

I've used all three of them. My favourite one is Maven, because:

a. **Convention over configuration**, which means developers don't need to create build process themselves (DRY).

b. It uses a standard directory layout and a default build lifecycle which enables me to automate the project's build infrastructure very quickly.

c. Easy to include dependencies (jars) via central maven repository or internal repository.

d. Many useful plugins to use in Maven;

e. Easy way of configuration via maven profiles or parameters;

f. Easy to plug into the continuous integration/delivery environment;

g. Easy to define different scope of each dependencies, so that the production jar/war file won't contain any test scope jars.

There're some other benefits of using Maven, I don't want to list all of them.

Gradle provides DSL in Groovy, so that I can inject logic in terms of how to build a project. Also, it provides the maven and ant plugin which is flexible to use. I used Gradle in some of my personal projects.

Ant is old school building tool, and I used it with Apache Ivy to achieve the similar result as Maven does. I have to define each build process as targets in the build.xml file.

13. When would you use a loop construct and when would you use the Streams API?

If I need to process a big list of data by such as filtering, grouping, using streams API is a better choice because it is fluent and easy to read. However, if I need to write a complex algorithm which needs nested loop on the data it would be better to use loop construct simply because of readability and easy to debug.

In addition, the performance of loop construct to do a sum of selected data in a big list is better than using map/reduce in stream APIs.

In conclusion, I would like to compromise by using both depending on different scenarios.

14. When and why do you write comments in your code?

When: it is REALLY necessary to write comments to my code. The following are the cases:

1. Java doc for methods in interfaces, for the sake of good readability in the contract between different components or systems.

2. Legacy code, if the code is ugly and not self-documenting or self-explanatory, and I'm being pushed by the management to delivery business values, I'll write comment for the code to explain what it does by appending to be refactored.

3. Describing a complex data structure and algorithm in a method so as to help other developers to understand my code.

4. Add "TODO...", so that my IDE can easily create outstanding tasks for me.

15. Name of the advantages of using source control management.

a. Source control is like a time machine, it stores all the history data of your code-base. For example, you don't have to save the commented-out code for future references.

b. Source control provide the branch feature so that multiple developers/testers can work on different branches in parallel without affecting each other (sometimes we need to resolve conflicts when merging code), which relatively boosts the productivity if it is used correctly.

c. Source control system like Git is the number 1 helper of continuous integration and test automation. For instance, I can setup Bamboo to deploy a specific branch to run the test cases designed for that specific branch.

16. What are the advantages/disadvantages of using enumerations rather than a bunch of String constants to represent the possible values of an attribute?

Approaches	Advantages	Disadvantages
String Constants	1. Can be used as static members within a class without defining another class. 2. Straightforward approach for a programmer.	1. Boilerplate code like "public static final String foo =..." 2. Not serializable because they belong to the class instead of instances. 3. It makes a class cumbersome. 4. Poor performance when used in if/switch because string content comparison.
Enum class	1. Enum is a class, multiple methods can be added into the enum class, and it can also implement interface(s). 2. values() method can return all the enum values, which is useful to be designed as a predicate to filter out invalid data in a list. 3. Serializable. 4. Better performance when Enum values are used in the if/switch within a big loop, because it compares the ordinal number instead of the String values. 5. Type safety.	1. Creating an extra class so as to define constants. 2. Not straightforward to a junior developer.

17. What is wrong with the following line of code:

```
verify(messageService).localiseException(any(), eq("exception.identity.change-default-
role.same-role"), "123456789", eq("Test"));
```

In Mockito, when a method to be stubbed has more than one argument, it is NOT possible to use ArgumentMatchers for only some of the arguments. In this case, you need to offer all arguments either by matchers or by exact values, that is to say, "12345678" should be changed to "any()" or "anyString()" or eq("12345678");

18. What are the various access specifiers in Java, and what visibility do they permit?

In Java there're four following access specifiers

1. public
2. private
3. protected
4. default (no specifier)

	Accessible/Visible inside the class	**Accessible/Visible inside the subclass inside the same package**	**Accessible/Visibl e outside the package**	**Accessible/Visible within the subclass outside the package**
default	yes	yes	no	no
private	yes	no	no	no
protected	yes	yes	no	yes
public	yes	yes	yes	yes