FPT Fresher Question Interview

JAVA CORE

1. Could you describe about "Strong typed"?
    - Check variables at compile time
    - Weak typed: check variables at runtime (script languages such as: JavaScript, PHP…).
2. What does "static" keyword mean?
    - Class resources
    - Used for method, attributes, inner class.
    - Available for all objects.
3. Describe the principles of OOPs.
    - Abstraction
    - Encapsulation
    - Inheritance
    - Polymorphism.
4. Explain about Polymorphism.
    - One name many forms
    - Override, overload methods.
    - Increase flexibility.
5. Explain about Inheritance.
    - Increase reusability
    - Extends class, implements interface.
    - Is – a relationship.
6. Explain about Encapsulation.
    - Hiding information and data.
    - Use access modifier(public, protected, private)
    - Make the system more modularized.
7. Explain about the different forms of Polymorphism?
    - Overriding
    - Overloading
    - Anonymous class.
8. What is the difference between method overloading and method overriding?

- Java method has five elements: modifiers, return types, names, parameters, exceptions.

a. Overloading method:
   - Same names
   - Others are flexible

b. Overriding:
   - Same names
   - Same parameters (number and type, order)
   - Access modifier is less restrict
   - Return type: same type or covariant type. (equal or narrower)
   - Exception: Checked exception (equal or narrower); flexible runtime exceptions.

9. What is dynamic binding?
   - Binding: Association btw reference and actual object.
   - Binding at runtime (Overriding method).
   - Static binding: at compile time.
10. Explain about "Abstraction"?
   - Increase extendability.
   - Increase abstraction of layered architecture.
   - Use interface or abstract class.
11. Could you explain "composition" and "inheritance" in JAVA?
   - Composition: Has – a relationship. (Famous example: Object Adapter pattern)
   - Inheritance: Is – a relationship. (Class adapter pattern).
12. Exception handling with composition and inheritance?
   - Inheritance:
     - An overriding method can throw any uncheck exceptions
     - An overriding method can throw narrower or fewer exceptions than the overridden method.
   - Composition:
     - Use try-catch block or throws exception when re-use method which throws exception
13. What are differences between abstract class and interface?
   - Implementation
   - Characteristics of method and attribute
   - Purpose of using.

a. Abstract class:

- Single inheritance with “extends” key word
- Could have both abstract and concrete methods. Attributes are normal as normal classes.
- Use when we want to have common behaviors for subclasses.

b. Interface:
- Support for multiple inheritance.
- Have only abstract methods
- Provide the contract.

14. What equals() and hashCode() method respond for?
- Equals() method:
  - Compare logically two objects.
- hashCode():
  - An integer number associcated with the objects using for storing and retriving in demands.
- Both methods are useful when we want to store objects in hash collection or set duplicate elements.

15. How and when override them?
- Equals() method:
  - Public boolean equals(Object obj){} (must pass Object type)
  - Check null -> check instanceof -> compare properties.
- Hashcode():
  - Public int hashCode(){}
  - Based on attributes we implement an agorithm to generate distinct numbers.

16. What is the difference between equals() and “==” ?
- equals(): Compare logically.
- “==” : Compare address.

17. What are differences between Comparator and Comparable?
- Comparable:
  - Override compareTo(Object obj)
- Comparator:
  - Act as the third party
  - Override compare(Object obj1, Object obj2)

18. Comparable interface? When to use them?"
- Comparable: implement to compare an object itself with another.
- Use:
  - Avoid duplicate elements on Set

    - Sort collections or array by using Collections.sort(collection) and Arrays.sort(array)

19. Is it possible to use multiple comparator?
    - Yes
    - With each criterion, we have implement Comparator interface
20. What is garbage collection? Can we enforce garbage collection to perform?
    - GC:
        - JVM mechanism for collecting unused objects and removing them.
        - Purpose: optimize and save memory.
        - Couldn't enforce but could register:
            - Object.finalize()
            - Call gc() method of System and Runtime.
21. What are differences between ArrayList and Vector?
    - ArrayList:
        - No synchronization
        - Increase 50% capacity.
    - Vector:
        - Synchronization
        - Double capacity when full size.
22. What are differences between HashMap and HashTable?
    - HashMap:
        - No synchronization
        - Allow one null key and many null values
    - HashTable:
        - Synchronization
        - Don't allow null key and null values.
23. What are differences between HashMap and TreeMap?
    - HashMap:
        - Don't guarantee the order of keys.
    - TreeMap:
        - Implements SortedMap interface
        - Order of keys is sorted.
24. How to make a Hashmap thread-safe?
    - Use ConcurrentHashMap
25. What are differences between List and Set?
    - List:

    - Support random access by index
    - Allow storing duplicate elements.
- Set:
    - Don’t support random access
    - No duplicate elements.

26. How to sort a list?
    - Implements Comparable -> Use Collections.sort();
    - Implements Comparator -> Use Collections.sort(list, comparator);

27. How to check duplicated elements in the Set?"
    - Override equals() and hashCode().
    - Wrong implementation of equals() can lead to memory leak problem.
28. How to find common elements in two sets?
    - Solution 1: Iterate two sets then check in loops one by one
    - Solution 2: Move elements to two lists then sort lists -> check common element with an efficient algorithm.
29. How to find + remove duplicated elements in a list?
    - Solution 1: Convert it to a set then set contains no duplicate objects.
    - Solution 2: Sort the list then compare continuous objects faster.

30. What is Iterator? How to use it?
    - A Java interface for traversing through collection.
    - hasNext(), next(), remove();
31. When you use Iterator?"
    - Traverse through a collection.
    - Make a copy of collection data.
    - No effects to the collection.
32. Can you explain TreeSet? HashSet?
    - TreeSet:
        - Implements SortedSet interface.
        - Use a tree for storage.
        - Elements are sorted.
    - HashSet:
        - Extends AbstractSet interface.
        - Use hash table for storage.
33. What are differences between Array and ArrayList?
    - Array:

    - Fixed size
    - Data type: primitive, objects.
    - Dimension: multi-dimension array.
- ArrayList:
    - Dynamic size.
    - Data type: only object.
    - Dimension: No.
    - Support Generics from Java 5.

34. How can we obtain an array from an ArrayList class?
    - ArrayList.toArray() (From ArrayList to Array)
    - Arrays.asList(array). (Vice-versa).
35. Have you ever worked with MultiMap?
    - MultiMap:
        - Component of Guava framework.
        - One key, multiple values.
        - get(key) return a list of values.
36. What's the LinkedList? When to use LinkedList?
    - LinkedList:
        - Provide linked list data structure.
        - Use large memory (for references).
        - Efficient for inserting or deleting.
        - Not efficient for random access as a normal list.
37. What are differences among String, StringBuilder and StringBuffer?"
    - Immutability:
        - String is immutable.
        - StringBuffer and StringBuilder are mutable.
    - Synchronization:
        - StringBuilder is not synchronized.
        - StringBuffer is synchronized

38. What meaning of String immutable? Can you explain the concept?"
    - When modifying a String, a new String object is created in memory, stored in the String pool and the instance refers to the new object.
39. Describe the basic steps to reverse a string?
    - Split a string into an array.
    - Use for loop to iterate the list from end to beginning.
40. What is Pass by Value and Pass by reference? Does Java support both of them?

- Pass by value:
    - Pass only the bit-pattern (copy) of value.
    - Method can't change the variable value.
- Pass by reference:
    - Receive a pointer of variable.
    - Java only supports Pass by value

41. What are differences between Deep copy and Shallow copy?
    - Deep copy:
        - Duplicate everything (Collection: structure + elements).
    - Shallow copy:
        - Copy as little as possible. (Collection: only structure + shared elements).
42. How do we implement Shallow cloning?
    - Implements Cloneable interface
    - Override clone().
43. How do we implement Deep cloning? (2 ways)
    - Solution 1: Implements Cloneable interface for all elements.
    - Solution 2: Serialization. (Serialize and deserialize).
44. Define exceptions?
    - Extends Exception class.
45. "Can you explain in short how JAVA exception handlings work?"
    - Use try-catch block, finally, "throws", "throw" keywords to handle exceptions.
    - Code in finally block always execute, use for cleaning code.
46. Can you explain different exception types?
    - Checked exception
        - Invalid condition out of program's control
        - Check at compile-time
    - Unchecked exeption
        - Check at run-time
        - Defects (bugs) in programs
47. What is the difference between error and exception?
    - Error:
        - Irrecoverable condition occurred at run-time
        - Can't repair at run-time
        - Eg: OutOfMemory
    - Exception:

    - Caused by bad input
    - Can handle
    - Eg: NullPointerException, IndexOutOfBoundException…

48. What is serialization?
    - Proccess to convert object to byte-stream for transferring through network or writing to disk.
49. How do we implement serialization actually?
    - Implement Serializable interface.
    - Use writeObject() and readObject() to serialize and deserialize
50. What's the use of Externalizable interface?
    - Purpose: to increase performance in some specific situations.
    - Use readExternal() and writeExternal() to read from stream and write object into stream.
51. What's difference between thread and process?
    - Thread:
        - Path of execution run on CPU, light weighted process
        - Related threads share same data memory
        - Have their own individual stacks
    - Process:
        - Collection of threads shared the same virtual memory
        - Every process has its own data memory location
52. What is thread safety and synchronization?
    - Thread safe:
        - A method that can run safely in multithread environment without any resource confliction.
    - Synchronization:
        - Assure resources (variable, object, method…) are not accessed by multiple threads at the same time
53. What is semaphore?
    - Object – helps one thread communicate with another to synchronize their operation
54. What is deadlock? How do you detect them? Do you handle them? And finally, how do you prevent them from occurring?
    - Lock: multiple processes access same resource at the same time
    - Deadlock: two thread waiting another in a cycle
55. How do we create threads? (2 ways)
    - Extends Thread class

- Implements Runable interface

56. What's difference between in using Runnable and Thread?"
    - Thread:
        - A class
        - Use when a class not extending another class
        - A thread has unique object instance associated with
    - Runnable:
        - An interface
        - Use when a class already extending another class
        - Many threads share the same object instance
57. How to implement thread safety? (2 ways)
    - Use “synchronized” with a block of code
    - Use “synchronized” with the method
58. "Let say we have 2 threads: A and B. Is there any way to be sure that thread A will execute before the thread B?"
    - setPriority() in Thread class
    - not guarantee A go first
59. Can you explain the wait() and notify() method?
    - wait()
        - A thread gives up its hold on the lock, goes to sleep
    - notify()
        - A thread wakes up and tries to acquire the lock again
60. How to monitor/manage threads? How to monitor JVM performance? JVM tuning?/ What tools do you use to check memory? "
    - Use JConsole and VisualVM
    - VisualVM:
        - Display real-time, high-level data
61. You run the application on Tomcat and run out of memory. What will you do?
    - Check log file
    - Use VisualVM to analyze
62. What is Stack and Heap Memory?
    - Heap:
        - Stores class instance + arrays
        - Shared memory
    - Non-heap:
        - ‘method area’
    - Stack memory:

    - Allocate automatic variable in function

63. How could you solve the memory leak?
    - Use good Java best practices
    - Consider static resources, set empty collections...
    - Minimize the variable scopes
    - Use tools to check before release applications
64. What will you do if your program has 500 Internal Server Error or OutOfMemoryException?
    - 500 error:
        - Check log file
        - Reprocedure and debug
    - OutOfMemory:
        - Check log file
        - Use tool to check memory leak
65. What's version of Java do you use? What are the notable characteristics of that version?
    - Java 6
    - Some features:
        - JAXB
        - Common annotations
        - …

XML

66. What is XML?
    - Extensible Markup Language.
    - Describe data.
    - Various programming languages support.
    - Checkable by XSD
    - Human readable in tags
67. How to validate the XML file?
    - Use XSD
68. What is XSD?
    - XML itself
    - Validate structure of XML file
    - Not mandatory
69. What is XSLT?
    - XSL: eXtensible Stylesheet Language for XML
    - XSLT: XSL Tranformation

  - Rule based language
  - Transform XML to other file formats (HTML, CS, RTF…)

70. Can you explain why your project needed XML?
    - Exchange data between 2 entities with same or different technologies but both understand XML
71. What is JAXB?
    - Java Architecture for XML Binding
    - Map java classes to XML
    - 2 main features:
      - Marshalling: Object -> XML
      - Unmarshalling: XML -> Object
72. Does JAXB support for SAX and DOM? (This question confuses me whether it is asking about parser or output document of JAXB)
    - About parser: http://stackoverflow.com/questions/9923326/does-jaxb-uses-sax-or-dom-internally
    - Output document: JAXB can marshal XML data to XML documents, SAX content handlers, and DOM nodes.

WEBSERVICE

73. What is Webservice?
    - method of communication between two electronic devices over the World Wide Web
74. What project did you use Webservice for?
75. What function did you Webservice provide?
76. What server did you use to run?
77. How did you test your Webservice?
    - RESTful client (add-on of Firefox)
    - SOAP UI
78. What do you use to parse JSON data?
    - Use Json or Jackson libraries
    - Create JSONObject (Please check sample code to see the details)
79. How do you read XML file using JAXB?
    a. Create POJO with annotations
    b. Create JAXBContext
    c. Create a marshaller or an unmarshaller to convert
    d. Please check sample code to see the details
80. How would you write a simple REST client?
    - Using Jersey:

    - Create client config
    - Create client
    - Get resource for client from URI
    - Getting data by path and media type/ posting data

81. How do you get the parameter in Restful?
    - Jersey: pathParam, QueryString
    - Spring MVC: @PathVariable
82. What XML Binding tool do you use?
    - JAXB
83. What are the differences between SOAP and REST?
    - Architecture:
        - SOAP: XML-based message protocol
        - REST: architectural style
    - Communication:
        - SOAP: WSDL
        - REST: XML + JSON (WADL)
    - Invocation:
        - SOAP: RPC method
        - REST: URL path
    - Returned result:
        - SOAP: doesn't return human readable
        - REST: return human readable (XML + JSON)
    - Protocol:
        - SOAP: HTTP, SMTP, FTP…
        - REST: HTTP
84. What is JAX-RS, and why did you use it ?
    - Java API for RESTful webservice
    - Support to create RESTful webservice
85. What JAX-RS implementation did you use ?
    - Jersey
    - Spring MVC
86. Could you explain about WADL?
    - Web Application Description Language
    - Machine-readable XML of HTTP-based web application
    - Models resources, relationship, methods applied, representation format
87. Have you worked with consuming or producing Web Service?
    - Both

- Jersey Client + Jersey

88. Can you explain about WSDL?
    - Webservice Description Language
    - XML-based interface for describing function
89. What are different states of object in Hibernate?
    - Transient: not associated with persistence context
    - Persistence: associated with persistence context
    - Detached: not associated with because persistence context is closed
90. What is the meaning of the Controller annotation?"
    - To identify that class acts as a controller

SPRING FRAMEWORK

91. What is Spring?
    - Open source framework, light weight
    - Layer architecture
    - Support java enterprise application
92. What are features of Spring?
    - Lightweight
    - IOC
    - AOP
    - Container
    - MVC
    - Transaction management
    - JDBC Exception Handling

93. What is IOC? Dependency Injection?
    - IOC:
        - Inversion of Control
        - Invert control of creating object from new operator to container
    - DI:
        - Dependency Injection
        - Implementation of IOC
        - All dependencies of an object are injected into it by framework
94. What is AOP?
    - Aspect Oriented Programming
    - Modularizes cross-cutting concerns (logging, security, transaction management..)
95. Explain Aspect, Advice, Joint Point, Pointcut?

- Aspect:
    - a modularization of a concern
    - cuts across multiple classes
    - Eg: transaction management
- Join point:
    - a point during the execution of a program
    - in Spring AOP: represents a method execution
- Advice:
    - action taken by an aspect at a particular join point
    - Different types: "around," "before" and "after" advice
- Pointcut:
    - Collection of Joint Points

96. What are different types of DI?
    - Constructor injection
    - Setter injection
    - Interface injection
    - Spring support Constructor Injection & Setter Injection
97. What are the benefits of DI?
    - Minimize amount of code
    - Make application more testable
    - Loose coupling
    - Eager instatiation + lazy loading
    - Flexible, security
98. Could you decribe the life cycle of Spring beans?
    - Bean Container finds definition of bean
    - Create a instance of bean
    - Depending on the interface, the properties of the bean -> setter method will be called
99. What is BeanFactory?
    - Base on Factory pattern and IOC design
    - Support 2 bean scopes: singleton + prototype
100. What is ApplicationContext? What is the different between BeanFactory and ApplicationContext?
    - ApplicationContext Derives from BeanFactory
    - Has all functionality of BeanFactory + support:
        - Internationalization messsages

  - Many enterprise service(EJB, JNDI…)
  - Access to resource (URL + file)
  - Application life-cycle events
  - Publish events to bean registered as listenter
  - Loading mutiple context

101. "How many types of bean scopes supported by Spring? And explain them."
    - 5 types:
      - Singleton: default scope of Spring, 1 object instance per Spring container
      - Prototype: new object is created + returned whenever you get the bean
      - Request: new object for each HTTP request
      - Session: new session is created -> new instance object of bean
      - Global session: same as HTTP session scope, applicable in portlet-based web app

JAVA DESIGN PATTERN

102. "What kind of design pattern that you know?"
    - Singleton
    - Factory + Abstract Factory
    - Service Locator
    - Façade
    - Observer
    - Builder
103. "What is façade pattern, factory pattern, singleton pattern? When you use them?"
    - Façade:
      - unified interface to a set of interface in as subsytem
      - hide complex system
    - Factory:
      - Create object without exposing instantiation logic to client
      - Refer newly created object through a common interface
    - Singleton:
      - A class which only one instance can be created, provides a global point of access this instance
104. What is Observer design pattern?
    - Defines one-to-many dependency between objects

- 1 object changes state => all of its dependences are notified and update automatically

105. "What is the service locator pattern?"
    - Encapsulate the processes involved in obtaining service with a strong abstraction layer
106. What is Builder design pattern? When should you use it?
    - Creational design pattern
    - Separate the construction of a complex object from its representation

DATABASE

107. What is outer join, left join, inner join?
    - Full outer join: Return all records from two tables.
    - Right outer join: Preserves the unmatched rows from first table (right), joining with Null rows in the shape of the second table (left).
    - Left outer join: Preserves the unmatched rows from first table (left), joining with Null rows in the shape of the second table (right).
108. What is index? When you should use index?
    - Data structure created in DB for finding data more quickly and efficiently.
109. What is transaction? Why do you use transaction?
    - A unit of work, performed against DB.
    - Two main purposes:
        - Allow correct recovery from failures and keep DB consistent even in cases of system failure.
        - Provide isolation btw programs accessing DB concurrently.

110. What is ACID?
    - Four main features of transaction: Atomicity, Consistency, Isolation, Durability.
    - Atomicity: Ensures that the entire sequence of operations is successfully or not.
    - Consistency: Ensures that the database properly changes states upon a successfully committed transaction.
    - Isolation: Enables transactions to operate independently of and transparent to each other.
    - Durability: Ensures that the result or effect of a committed transaction persists in case of a system failure.

111. Can you explain database partitioning?
    - A division of a logical DB or its constituting elements into distinct independent parts.
    - Purposes:
        - For manageability
        - Performance
    - Three common criteria to split DB:
        - Range partitioning (use certain range)
        - List partitioning (assign a list of values)
        - Hash partitioning (use value of a hash function)

112. What is the difference between DELETE and TRUNCATE?
    - DELETE:
        - Deletion of each row or the whole table gets logged and physically deleted.
        - Could have condition, triggers
        - Need to commit the changes
    - TRUNCATE:
        - Log the de-allocation of the data pages in which the data exists.
        - No need the commit statement.
        - Don’t have condition, triggers.
113. When to use "group by" clause?
    - Group similar data
114. What is the different between "Having" and "Where" clause?
    - Having: conditions used with group by.
    - Where: conditions used with SELECT.
115. What is a Sub Query?
    - A query nested inside a SELECT statements
    - Alternative to complex join statements.
116. What is a View?
    - A virtual table created on basis of the result set returned by the select statement.
    - Increase performance and security
117. How to find how many duplicated record in a table?
    - Step1: count all distinct records in table.
    - Step2: count all records in that table
    - Step3: The result: the result in step2 subtracts to the result in step 1.

118. How to count a number of records in a table.
    - Use count(*) from Table.
119. What is composite key?
    - One primary key consists of two tables.

HIBERNATE

120. What the main advantages of ORM are like hibernate?
    - Productivity
    - Maintenance
    - Performance
    - Portability
121. How to make entity from a class?
    - Use annotation @Entity in that class.
122. What are the core interfaces of Hibernate framework?
    - Session interface:
        - Basic interface for all hibernate apps
        - Light weighted.
    - SessionFactory interface:
        - Only one SessionFactory
        - Shared by all the application threads.
    - Configuration interface:
        - Configure bootstrap action
    - Transaction interface:
        - Optional interface
        - Abstract the code from a transaction implemented such as JDBC/JTA.
    - Query and Criteria interface:
        - Queries from user are allowed by this interface.
123. What is Hibernate proxy? Explain how to configure Hibernate.
    - Mapping of classes can be made into a proxy instead of a table.
    - A proxy is returned when actually a load is called on a session.
    - Contains actual method to load the date.
    - Is created by default by Hibernate
124. Explain the Collection types in Hibernate.
    - A collection is defined as a one-to-many reference
    - The simplest type is <bag>: list of unordered objects and can contain duplicates. (Similar to List).

125. What is lazy fetching in hibernate?
    - Decide whether to load child objects while loading the Parent Object.
    - Can be done by a setting in hibernate mapping file of the parent class.Lazy = true;
126. Could you explain when to cause lazy loading exception?
    - Try to get elements from collection, working outside the Hibernate session.
    - Session was closed before getting lazy detached collection.
    - To avoid:
        - Check the code which operates with collection executed within transaction.
        - Mark method with @Transactional.
127. What is Hibernate Query Language (HQL)?
    - Similar to but shorter SQL, is query language for Hibernate.
    - Instead of operating on tables and columns, working with object.
128. Explain the general flow of Hibernate communication with RDBMS?
    - Load Hibernate configuration file and create configuration object. (Automatically load all hbm mapping file).
    - Create session factory from configuration file.
    - Create session from session factory
    - Create HQL query.
    - Execute query to get list containing Java objects.
129. Could you explain how to deal with Hibernate concurrency?
    - Two types:
        - Optimistic way: Use version annotation
        - Pessimistic way: use setLockMode method.
130. How many levels of cache in Hibernate?
    - 2 levels:
        - First-level cache: associates with Session object. (Hibernate uses this by default).
        - Second-level cache associates with SessionFactory object.
131. How many types of transaction in Hibernate?
    - 2 types:
        - Managed: use container to manage.
        - Un-managed: manage by your own.
132. What is JPA?
    - Java Persistence API

- The entity persistence model for EJB3.0.
- Standardized persistence framework which is implemented by Hibernate (TopLink…).
- EntityManager provides vendor independent access to persistence.
- Use JQL.

133. Explain the advantages of JPA? Explain the general flow of Hibernate JPA communication with RDBMS?
    - Is standard
    - Not tie to you to Hibernate.
    - Give you most of features of Hibernate except:
        - Doesn't have Hibernate's DeleteOrphan cascade type.
    - The general flow of Hibernate JPA communication with RDBMS:
        - Load Hibernate configuration file and create configuration object. (Automatically load all hbm mapping file).
        - Create session factory from configuration file.
        - Create session from session factory
        - Create HQL query.
        - Execute query to get list containing Java objects.
134. What is EJB? What are the advantages of using EJB?
    - Enterprise Java Bean
    - Server side component written in Java Language.
    - Replicate the table model as objects.
135. How many kinds of EJB?
    - 3 kinds of EJB:
        - Entity Bean
        - Session Bean
        - Message-driven Bean.
136. How many Message models? Step to create a message-driven bean?
    - 2 models
        - Publishers - Subcribers
        - Point To point.
    - Step by step to create message-driven bean (Em chưa làm).
137. How do you decide when you should you session, entity or message-driven bean?
    - Entity Bean:
        - Are data objects
        - Represent persistent data

    - Responsible for DB CRUD.
- Session Bean:
    - Only implement business logic and work flow.
- Message-driven beans:
    - Receiving asynchronous messages from other systems.

138. Can you compare Stateless and stateful session bean?
    - Stateful Session Bean:
        - Are retained during working session.
        - Lifecycle (postConstruct and preDestroy)
    - Stateless Session Bean:
        - Are not retained for each client request
        - Container can assign the same bean for different clients.
        - Lifecycle (postConstruct, preDestroy, prePassive, postActivate).

LINUX

1. How to read log file?
    a. TAIL or HEAD commands
2. How to set environment variable?
    a. EXPORT or update the profile files
3. How to execute file or run script in Linux?
    a. Use sh statement