**[MODULE-2] Audit Module 2 - JavaCore**

##Nội dung câu hỏi và phần trả lời

**[ Câu hỏi ] 1. Could you describe about Strong typed?**

- Check variables at compile time

- Weak typed: check variables at runtime (script languages such as: JavaScript, PHP…).

**[ Câu hỏi ] 2. What does static keyword mean?**

- Class resources

- Used for method, attributes, inner class.

- Available for all objects.

**[ Câu hỏi ] 3. Describe the principles of OOPs.**

- Abstraction

- Encapsulation

- Inheritance

- Polymorphism.

**[ Câu hỏi ] 4. Explain about Polymorphism.**

- One name many forms

- Override, overload methods.

- Increase flexibility.

**[ Câu hỏi ] 5. Explain about Inheritance.**

- Increase reusability

- Extends class, implements interface.

- Is – a relationship.

**[ Câu hỏi ] 6. Explain about Encapsulation.**

- Hiding information and data.

- Use access modifier(public, protected, private)

- Make the system more modularized.

**[ Câu hỏi ] 7. Explain about the different forms of Polymorphism?**

- Overriding

- Overloading

- Anonymous class.

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 9. What is dynamic binding?**

- Binding: Association btw reference and actual object.

- Binding at runtime (Overriding method).

- Static binding: at compile time.

**[ Câu hỏi ] 10. Explain about Abstraction?**

- Increase extendability.

- Increase abstraction of layered architecture.

- Use interface or abstract class.

**[ Câu hỏi ] 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).

**[ Câu hỏi ] 12. Exception handling with composition and inheritance?**

- Inheritance:

o An overriding method can throw any uncheck exceptions

o An overriding method can throw narrower or fewer exceptions than the overridden method.

- Composition:

o Use try-catch block or throws exception when re-use method which throws exception

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 14. What equals() and hashCode() method respond for?**

- Equals() method:

o Compare logically two objects.

- hashCode():

o 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.

**[ Câu hỏi ] 15. How and when override them?**

- Equals() method:

o Public boolean equals(Object obj){} (must pass Object type)

o Check null -> check instanceof -> compare properties.

- Hashcode():

o Public int hashCode(){}

o Based on attributes we implement an agorithm to generate distinct numbers.

**[ Câu hỏi ] 16. What is the difference between equals() and “==” ?**

- equals(): Compare logically.

- “==” : Compare address.

**[ Câu hỏi ] 17. What are differences between Comparator and Comparable?**

- Comparable:

o Override compareTo(Object obj)

- Comparator:

o Act as the third party

o Override compare(Object obj1, Object obj2)

**[ Câu hỏi ] 18. Comparable interface? When to use them?**

- Comparable: implement to compare an object itself with another.

- Use:

o Avoid duplicate elements on Set

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

**[ Câu hỏi ] 19. Is it possible to use multiple comparator?**

- Yes

- With each criterion, we have implement Comparator interface

**[ Câu hỏi ] 20. What is garbage collection? Can we enforce garbage collection to perform?**

- GC:

o JVM mechanism for collecting unused objects and removing them.

o Purpose: optimize and save memory.

o Couldn’t enforce but could register:

♣ Object.finalize()

♣ Call gc() method of System and Runtime.

**[ Câu hỏi ] 21. What are differences between ArrayList and Vector?**

- ArrayList:

o No synchronization

o Increase 50% capacity.

- Vector:

o Synchronization

o Double capacity when full size.

**[ Câu hỏi ] 22. What are differences between HashMap and HashTable?**

- HashMap:

o No synchronization

o Allow one null key and many null values

- HashTable:

o Synchronization

o Don’t allow null key and null values.

**[ Câu hỏi ] 23. What are differences between HashMap and TreeMap?**

- HashMap:

o Don’t guarantee the order of keys.

- TreeMap:

o Implements SortedMap interface

o Order of keys is sorted.

**[ Câu hỏi ] 24. How to make a Hashmap thread-safe?**

- Use ConcurrentHashMap

**[ Câu hỏi ] 25. What are differences between List and Set?**

- List:

o Support random access by index

o Allow storing duplicate elements.

- Set:

o Don’t support random access

o No duplicate elements.

**[ Câu hỏi ] 26. How to sort a list?**

- Implements Comparable -> Use Collections.sort();

- Implements Comparator -> Use Collections.sort(list, comparator);

**[ Câu hỏi ] 27. How to check duplicated elements in the Set?**

- Override equals() and hashCode().

- Wrong implementation of equals() can lead to memory leak problem.

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 30. What is Iterator? How to use it?**

- A Java interface for traversing through collection.

- hasNext(), next(), remove();

**[ Câu hỏi ] 31. When you use Iterator?**

- Traverse through a collection.

- Make a copy of collection data.

- No effects to the collection.

**[ Câu hỏi ] 32. Can you explain TreeSet? HashSet?**

- TreeSet:

o Implements SortedSet interface.

o Use a tree for storage.

o Elements are sorted.

- HashSet:

o Extends AbstractSet interface.

o Use hash table for storage.

**[ Câu hỏi ] 33. What are differences between Array and ArrayList?**

- Array:

o Fixed size

o Data type: primitive, objects.

o Dimension: multi-dimension array.

- ArrayList:

o Dynamic size.

o Data type: only object.

o Dimension: No.

o Support Generics from Java 5.

**[ Câu hỏi ] 34. How can we obtain an array from an ArrayList class?**

- ArrayList.toArray() (From ArrayList to Array)

- Arrays.asList(array). (Vice-versa).

**[ Câu hỏi ] 35. Have you ever worked with MultiMap?**

- MultiMap:

o Component of Guava framework.

o One key, multiple values.

o get(key) return a list of values.

**[ Câu hỏi ] 36. What's the LinkedList? When to use LinkedList?**

- LinkedList:

o Provide linked list data structure.

o Use large memory (for references).

o Efficient for inserting or deleting.

o Not efficient for random access as a normal list.

**[ Câu hỏi ] 37. What are differences among String, StringBuilder and StringBuffer?**

- Immutability:

o String is immutable.

o StringBuffer and StringBuilder are mutable.

- Synchronization:

o StringBuilder is not synchronized.

o StringBuffer is synchronized

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 40. What is Pass by Value and Pass by reference? Does Java support both of them?**

- Pass by value:

o Pass only the bit-pattern (copy) of value.

o Method can’t change the variable value.

- Pass by reference:

o Receive a pointer of variable.

o Java only supports Pass by value

**[ Câu hỏi ] 41. What are differences between Deep copy and Shallow copy?**

- Deep copy:

o Duplicate everything (Collection: structure + elements).

- Shallow copy:

o Copy as little as possible. (Collection: only structure + shared elements).

**[ Câu hỏi ] 42. How do we implement Shallow cloning?**

- Implements Cloneable interface

- Override clone().

**[ Câu hỏi ] 43. How do we implement Deep cloning? (2 ways)**

- Solution 1: Implements Cloneable interface for all elements.

- Solution 2: Serialization. (Serialize and deserialize).

**[ Câu hỏi ] 44. Define exceptions?**

- Extends Exception class.

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 46. Can you explain different exception types?**

- Checked exception

o Invalid condition out of program’s control

o Check at compile-time

- Unchecked exeption

o Check at run-time

o Defects (bugs) in programs

**[ Câu hỏi ] 47. What is the difference between error and exception?**

- Error:

o Irrecoverable condition occurred at run-time

o Can’t repair at run-time

o Eg: OutOfMemory

- Exception:

o Caused by bad input

o Can handle

o Eg: NullPointerException, IndexOutOfBoundException…

**[ Câu hỏi ] 48. What is serialization?**

- Proccess to convert object to byte-stream for transferring through network or writing to disk.

**[ Câu hỏi ] 49. How do we implement serialization actually?**

- Implement Serializable interface.

- Use writeObject() and readObject() to serialize and deserialize

**[ Câu hỏi ] 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.

**[ Câu hỏi ] 51. What's difference between thread and process?**

- Thread:

o Path of execution run on CPU, light weighted process

o Related threads share same data memory

o Have their own individual stacks

- Process:

o Collection of threads shared the same virtual memory

o Every process has its own data memory location

**[ Câu hỏi ] 52. What is thread safety and synchronization?**

- Thread safe:

o A method that can run safely in multithread environment without any resource confliction.

- Synchronization:

o Assure resources (variable, object, method…) are not accessed by multiple threads at the same time

**[ Câu hỏi ] 53. What is semaphore?**

- Object – helps one thread communicate with another to synchronize their operation

**[ Câu hỏi ] 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

**[ Câu hỏi ] 55. You run the application on Tomcat and run out of memory. What will you do?**

- Check log file

- Use VisualVM to analyze

**[ Câu hỏi ] 56. What is Stack and Heap Memory?**

- Heap:

o Stores class instance + arrays

o Shared memory

- Non-heap:

o ‘method area’

- Stack memory:

o Allocate automatic variable in function

**[ Câu hỏi ] 57. 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

**[ Câu hỏi ] 58. What will you do if your program has 500 Internal Server Error or OutOfMemoryException?**

- 500 error:

o Check log file

o Reprocedure and debug

- OutOfMemory:

o Check log file

o Use tool to check memory leak

**Mục tiêu**

* Audit Module 2 - JavaCore

**URL fill kết quả audit**

* <https://docs.google.com/spreadsheets/d/1cRw6Aqou1YGZfaCjn2HHv4pUQrunx3_G/edit#gid=105659381>

**Youtube Link**

* [[Case Study][Module 2]Hướng dẫn cài đặt Intelliji](https://www.youtube.com/watch?v=qvsYyTcSulA&list=PL9yu4ScFhKGyYTjikvXLxpTn2RMJZw3PW)
* [[Java core][Module 2]Override và Overload](https://www.youtube.com/watch?v=soF6VeampxY&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=3)
* [[Java core][Module 2]Access Modifier](https://www.youtube.com/watch?v=qzfMZigVMDc&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=2)
* [[Java core][Module 2]Các tính chất của OOP](https://www.youtube.com/watch?v=0F_8a5_fKno&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=4)
* [[Java core][Module 2]Abstract](https://www.youtube.com/watch?v=t9in5g6vsSg&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=6&t=0s)
* [[Java core][Module 2]Interface](https://www.youtube.com/watch?v=rUFUgrkMg4o&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=6)
* [[Java core][Module 2]Inheritance](https://www.youtube.com/watch?v=2Z0_WQCjIB4&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=4)
* [[Java core][Module 2]Abstract vs Interface](https://github.com/codegymdanang/CGDN-Java-InterviewQuestion/blob/master/Module_2_Core)
* [[Java core][Module 2]Ép kiểu ngầm định và tường minh](https://www.youtube.com/watch?v=VQjlRs0aiUQ&feature=youtu.be)
* [[Java core][Module 2]Bộ nhớ Heap vs Stack](https://youtu.be/werAdblsT1s)
* [[Java core][Module 2]Comparable vs Comparator](https://www.youtube.com/watch?v=v_rcgxYK7gY&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=8)
* [[Java core][Module 2]Array và ArrayList](https://youtu.be/Qz5pyXdWTbQ)
* [[Java core][Module 2]Generic](https://youtu.be/2ADRQUlsRtI)
* [[Java core][Module 2]Cài đặt List Part 1](https://youtu.be/qmZIGtWtMHI)
* [[Java core][Module 2]Cài đặt List Part 2](https://youtu.be/wMS091Df1b0)
* [[Java core][Module 2]Cài đặt LinkedList](https://youtu.be/FOdevD5rfcA)
* [[Java core][Module 2]OOD](https://youtu.be/CDw7-A5xohM)
* [[Java core][Module 2]Clean code](https://youtu.be/aSDKxAUqcTc)
* [[Java core][Module 2]Exception](https://youtu.be/2BBAVtiks1w)
* [[Java core][Module 2]Thread](https://youtu.be/uc_2gv1qJOw)
* [[Java core][Module 2]IO File](https://www.youtube.com/watch?v=WOOdgFiC_S4&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0)
* [[Java core][Module 2]Regular Expression](https://www.youtube.com/watch?v=Sj-RRjiyv-Q&list=PL9yu4ScFhKGzZVSeCVGH2I2xMuYBrNYt0&index=7)
* [[CaseStudy][Java core][Module 2]](https://www.youtube.com/watch?v=ek4A-3iImug&list=PL9yu4ScFhKGyYTjikvXLxpTn2RMJZw3PW&index=2)