JAVA / JAVA 8 / Design Patterns

1. What is the output?
2. **import** java.util.\*;
3. **public** **class** Test2{
4. **public** **static** **void** main(String a[]){
5. Map s = **new** Hashtable();
6. s.put(**null**,**null**);
7. System.out.println(s);
8. }
9. }
10. null
11. NullPointerException
12. [null = null]
13. []

2) What is the output?

1. **import** java.util.\*;
2. **public** **class** Test4{
3. **public** **static** **void** main(String a[]){
4. Map s = **new** LinkedHashMap();
5. s.put("1","one");
6. s.put("3","three");
7. s.put("2","two");
8. System.out.println(s);
9. }
10. }
11. [1=one,2=two,3=three]
12. NullPointerException
13. [1=one,3=three,2=two]
14. []

3) How will you synchronize below given HashMap object?

HashMap hashmap = new HashMap();

1. Collections.synchronized(hashmap);
2. Map.synchronizedMap(hashmap);
3. Collections.synchronizedMap(hashmap);
4. Map.synchronized(hashmap);

4) Which is a valid declaration within an interface?

1. protected short stop = 23;
2. public Boolean demo(long bow);
3. final void demo(short stop);
4. static char demo(double duty);

5) Can an abstract class define both abstract methods and non-abstract methods?

1. No-it must have all one or the other.
2. Yes-but the child classes do not inherit the abstract methods.
3. No-it must have all abstract methods.
4. Yes-the child classes inherit both.

6) For a given code snippet

1. **public** **class** Constructor {
2. **public** Constructor (**int** x, **int** y, **int** z)
3. {}
4. }

Which of the following is considered as overloaded constructor?

1. public void Constructor(int x, int y, byte z){}
2. private Object Constructor() {}
3. Constructor() {}
4. protected int Constructor(){}

7) Lambda expressions in java 8 are based on

1. Procedural programming
2. Functional programming
3. Data programming
4. All

8) How many methods are there in functional interface in Java 8?

1. 0
2. 1
3. 2
4. 3

9) Stream operations in java 8 can be divided into

1. Terminal types
2. Intermediate types
3. All
4. None

10) Which of these represents a process that accepts one argument and produces a result in Java 8

1. Function
2. Process
3. Method
4. JavaFunctions

11) In java 8 Function is ?

1. Class
2. Interface
3. Lambda Expression
4. Object

12) What can help us in avoiding NullPointeExceptions and null checks in java 8

1. Optional
2. Required
3. NotNull
4. NotRequired

13) code before Java 8 essentially used to be

1. Declarative
2. Imperative
3. Subjective
4. None

14) In java 8, R apply(T t) is a method of-

1. Function
2. Process
3. Predicate
4. None

15) Which of these does Stream map() operates on

1. Class
2. Interface
3. Predicate
4. Function

16) Which method can be used to check null on an Optional variable in Java 8

1. isPresent()
2. isNullable()
3. isPresentable()
4. isNotNull()

17) Which of these does forEach() operates on

1. Methods
2. Consumer
3. Producer
4. Predicate

18) `map` and `filter` are

1. Parallel operations
2. Intermediate operations
3. Terminal operations
4. Initial operations

19) Lambda expressions used in Streams should be

1. Stateful
2. Stateless
3. Either stateful or stateless
4. Stateful only if they are thread-safe

20) Which design pattern creates a complex object using simple objects and using a step by step approach?

1. Facade Pattern
2. Builder pattern
3. Factory pattern
4. Iterator Pattern

21) Adapter pattern is also called \_\_\_\_\_\_\_\_\_\_.

1. Decorator pattern
2. Template pattern
3. Wrapper pattern
4. State pattern

22) To make user use same instance of class across application we should use \_\_\_\_\_\_\_\_\_\_.

1. Iterator Pattern
2. Command Pattern
3. Facade Pattern
4. Singleton Pattern

23) Which of the following is true for facade pattern?

1. The facade class provides basic, simplified services to clients by taking upon itself the job of dealing with a complex sub-system.
2. The Facade pattern eases interaction between a client and a sub-system of suppliers by providing a simpler interface to the sub-system
3. All of the above
4. Sub-systems may be complex, but often clients only need basic services that can be supplied through a simple interface.