Concepts: static methods, static attributes and static initialization blocks.

Theory points

1. By default java command executes main method from a class whichever you are running.
2. If main method is not available, then you get runtime error.
3. main method always should be

```
public static void main(String[] args)
{
}

Or

public static void main(String[] p)
{
}
```

Or

```
static public void main(String[] x)
{
}
```

4. java command executes only main method.
5. class can have any number of methods. But java command executes only main method
6. we can develop any number of an un used methods inside a class.
7. Even un used method also should be syntactically correct to compile the whole file.
8. We should call test method from a main method to execute test method.
9. We can call a method any number of times.
10. Methods are mainly for developing reusable code.
11. We can call test1 method from the main method any number of times.

12. We can call test2 method from test1 method any number of times.
13. We can call test1 method from main method and test2 method from test1 method
14. Calling test1 method from test2 method and test2 method from test1 method gives runtime error (StackOverFlowError)
15. We can't call an un defined methods. We can call only defined methods. We cant use an un defined methods.
16. Method can take any number of arguments.
17. Method arguments can be any datatype.
18. Method arguments are local to the same method.
19. One method argument cant be used in another method.
20. Calling statement of a method should supply values to all the arguments.
21. Calling statement should supply a value of same type of an argument

22. Calling statement should supply values to every argument and in the same order by ensuring value type is same as argument type.
23. We cant supply value as a boolean if method argument type is an int type.
24. Two arguments of a same method can have same datatype.
25. Two arguments of a same method should not have same name.
26. Local variable name should not have a same name of any argument of the same method.
27. Argument of a method can be used directly as its getting initialized with a calling statement of a same method.
28. Arguments re initializing in the method body is not a meaningful way of the design.
29. Method should have a return type.
30. Return type of a method should be before method name while defining a method.
31. There are 3 Return types available to method.

1. void
2. Any primitive data type
3. Any derived data type

32. void means nothing. It should not return a value.

33. If method return type as a void, then
    1. Method should not return any value
    2. return statement is not a mandatory.
    3. method can have optional return statement without a value
    4. return statement with a value is not possible

34. if you want to keep a return statement even though method return type is void, then keep a return statement without a value and ensure return statement is a last statement in the current block.

35. If something went wrong inside a method, then skip the remaining portion of a method by keeping return statement explicitly.

36. if return type of a method is other than void, then return statement with a value is a mandatory.
37. if return type of a method is any primitive or any derived, then return statement with a value is a mandatory.
38. If any data type is already defined inside a programming language, then its called as a primitive data type.
39. Java primitive data types:
    1. byte
    2. short
    3. int
    4. long
    5. float
    6. double
    7. boolean
    8. char

40. If method return type is an int type then return statement with an int value is mandatory

41. If method return type is boolean type then return statement with a boolean value is mandatory
42. Every class type or interface type is a derived data type.
43. String is a derived data type. String can be a return type to a method.
44. Any class type can be a return type to a method.
45. Any interface type can be a return type to a method.
46. Default value to any derived data type is null.
47. static attributes are initializing with the default values.
48. Default value for any attribute is depending on the data type.
49. Default values of byte, short, int and long is 0 (zero)
50. Default values of float and double is 0.0
51. Default value of boolean is false

52. Default value of String or any class type (or even interface type) is null
53. Default value of any array type is null
54. Two attributes should not be with the same name even though data type is changing.
55. Attribute name and local variable name both can have a same name
56. If attribute and local variable both are with the same name, then local variable will have more preference.
57. Use classname to refer an attribute even though local variable name is same as an attribute.
58. If any attribute of a class initializing while declaring itself, then that statement is called as an initializer.
59. From the initializer, we can call a method which is having return type as attribute data type.
60. While using any class first name, it will load to the memory.

1. Store static members in the memory with the default values
2. JVM notifying / executing all initializers
3. Ready to use

61. All static members are class members. All static members are loading to the memory while loading a class. class loads to the memory only one time and it is while class using the first time. We can use static members with a class name.
62. While using static member of one class in another class, you should use along with the class name.
63. All static initializers are executing only one time and it is while class is loading to the memory.
64. All static initializers are executing from top to bottom in the class while class is loading to the memory.
65. All static initializers are executing before main method execution

66. If you want to execute main method more than one time, then develop a calling statement inside an initializer.
67. Direct read of any static attribute is not possible before JVM notifying.
68. If there is a direct read before JVM notifying of any attribute, the Illegal Forward Reference (IFR) compilation error.
69. Indirect read or any type of write is allowed even in before JVM notifying.
70. We can develop any number of classes in the same file.
71. If more than one class developed in the same class, then the maximum one class can be public. More than one class cant be public in the same file.
72. If file containing a public class, then file name should be public class name only.
73. If file doesn't have any public classes, then file name can be any file name.

74. While compiling a java file, class file will be generating for every class. If java file containing 10 classes, then compiler generates 10 class files.
75. We can use a class of one java file in another java file. In this case, try compiling both the java files together with a space as a separator.
76. Let us consider class Person developed inside a Person.java and used inside a Manager.java, then you can compile straight away Manager.java. While compiling Manager.java, even Person.java also compiling automatically.
77. Let us consider class Person developed inside a Test.java and used inside a Manager.java, then you cant compile straight away Manager.java. While compiling Manager.java, you should supply even Test.java also. Like

```
src>javac -d ../classes Manager.java Test.java
```

78.
79.

FAQS

1. Which method executes java command by default.
2. Is it possible to run a class which doesn't have a main method
3. Is it possible to compile a class, if it doesn't have main method?
4. Is it possible to develop an un used methods inside a class?
5. Whether an un used method compiles or not?
6. How to execute test method?
7. Is it possible to call same method again and again?
8. What we will achieve through methods?
9. When we get StackOverFlowError?
10. Is it possible to have two arguments of a same method with same datatype????
11. Is it possible to develop a method without return type?
12. When method can be developed without return statement?

13. When return statement is required even though return type is a void?
14. If something went wrong inside a method, then how to skip the remaining portion of a method??
15. When return statement with a value is a mandatory inside a method??.
16. What is a primitive data type?
17. What are the primitive data types available?
18. What are the derived data types?
19. What is the default value to the derived data type?

20. What is a default value of a boolean attribute?
21. What is a default value to an array type?
22. If attribute and local variable both are with the same name, then which one will have more preference.
23. What is an initializer???

24. Is it possible to call a method from an initializer???
25. When class will load to the memory.
26. When static initializers will executes?
27. Is it possible to execute some thing before main method???
28. Is it possible to execute main method more than one time???
29. Which read of any static attribute is not possible before JVM notifying??.
30. When Illegal Forward Reference (IFR) compilation error.
31. If more than one class developed in the same class, then the how many classes can be public?
32. If java file containing 10 classes, then how many class files will be generating?.
33. We can use a class of one java file in another java file. In this case, try compiling both the java files together with a space as a separator.

34. Let us consider class Person developed inside a Person.java and used inside a Manager.java, then is it possible to compile straight away Manager.java?