**Index**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **Topic** | **Page No.** |
| 1. | **Theory on static** | 1-3 |
| 2. | **Global Variables** | 4-10 |
| 3. | **Initializers** | 11-19 |
| 4. | **SIB** | 19-23 |
| 5. | **Multiple classes of same requirement developing in the same file** | 24-32 |

|  |  |  |
| --- | --- | --- |
| 1. | **What is a global variable?**  A variable declared within a class can be called as a global variable.  Also called as field, property, attribute. It can be used throughout the class.  All global variables are automatically initialized with the default values. Default values are depending on the datatype.  Global variables can be used straightaway without initialization. | |
| 2. | **How many types of global variables are there in Java?**  There are two types of global variables in Java :   1. Static global variable 2. Non-static global variable | |
|  | **Can there be a local and a global variable in a class with same name and same datatype?**  Yes, A class can have a local as well as global variable with the same name and same datatype.  When local and global variable are having same name, local variable will have higher preference in the block. | |
| 4 | **Can there be a local and a global variable in a class with same name and different datatypes?**  Yes, A class can have a local as well as global variable with the same name and same datatypes. | |
| 5 | **Can more than one global variable of different/same datatype have same name?**  No, In a class each global variable should have unique name whether they are of same or different datatype. | |
| 6 | **Which are different ways to access global variable?**  There are 2 different ways to access static global variables:   1. Straightaway : globalVariableName 2. Along with the class name : className.globalVariableName   It is advisable to use the 2nd way to avoid chances of modifying same named local variable by mistake.  To access non-static global variable :   1. Inside non-static block inside same class 🡪 using nonStaticVar/ referenceVar.nonStaticVar 2. Inside static block in same or different class & Non-static block in another class 🡪 using referenceVar.nonStaticVar.   [note : nonStaticVar🡪 non static global variable, referenceVar🡪 reference variable, className🡪 name of class] | |
| 7 | **Describe necessary points for static global variable/ class members.**  Every static member is called as a class member.  While loading a class, static members are loaded into memory.  To use static member use it with a class name. | |
| 8. | **How many times do static members get loaded into the memory?**  Static members will be loaded into memory only once during execution of program.  While loading a class into memory static members get loaded. | |
| 9. | | **How many times a class gets loaded into memory?**  Class is loaded into memory only once; it gets loaded during the first use and is  available till program execution completed. | |
| 10. | | **When does static initializer(s) get executed?**  Static initializer(s) are executed when the class is loaded into memory. | |
| 11 | | **State the order in which the initializers get executed.**  Initializers are executed from top-to-bottom. | |
| 12 | | **What is the purpose of static initialization block (SIB)?**  Certain code which should be executed only once inside a class for entire execution then static initialization block is required.  Eg. Initialization of global static variable with the value which is available in the file system.  Code to connect with the file system, get the value from file, assigning value to global static variable should be placed in SIB. | |
| 13 | | **Which members are also called as class members?**  Every static member of a class is called as a class member. | |
| 14 | | **How to use static member of class A in class B?**  A.staticMemberName | |
| 15 | | **Is it possible to develop multiple classes in one .java file?**  Yes, a .java file can contain any number of classes. | |
| 16 | | **While developing multiple classes in same .java file. What should be the filename?**  If a .java file contains a public class then filename should be same as public className.  Otherwise, filename can be any. | |
| 17 | | **In a .java file, how many public class(es) can be there?**  In a .java file, there can be maximum 1 public class.  If in a .java file more than 1 public classes are there 🡪compile time error | |
| 18 | | **How many .class files will be generated for 1 .java file?**  While compiling .java file depending on the number classes a java file contains .class  files will be generated.  Eg. If a java file contains 10 classes(1 public class , 9 classes) 🡪 10 .class files  If a java file contains 1 class 🡪 1 .class file | |
| 19 | | **State condition to run a class successfully.**  In order, to run a class successfully it should contain a main() method. | |
| 20 | | **If a class contains static initializer and a main() method which executes first?**  Static initializer | |
| 21 | | **What happens if a java file contains 1 class with several static initializer blocks but no main() method?**  Compilation will be successful.  In JDK1.7, on trying to run the program it results Runtime Error. Because on trying to  run a class, before loading a class the JVM checks whether main() method is available in  the class specified for running. As there is no main() method, class wont be loaded into  memory and thus SIB will not be executed.  In JDK1.6, on trying to run the program it doesn’t require main() method to load the class into memory. So it will execute the SIB. | |
| 22 | | **Comparison JDK1.6 and JDK1.7 with respect to main() method.**   1. For JDK1.6 if class doesn’t contains main() method then you will get runtime 2. exception saying no such method error : NoSuchMethodError. Whereas in JDK1.7 it provides more meaningful error information: Main method not found in class \_\_\_, please define main method as public static void main(String[] args) 3. If a class contains only static block,   In JDK1.6, static block is executed then it looks for main() method. Whereas in JDK1.7, to run a java program main() method is mandatory. Even though a class contains static blocks they wont be executed if a class doesn’t contain main() method.   1. If the class contains main() method whether it is JDK1.6 or JDK 1.7 there is no change in execution sequence | |
| 23 | | **Suppose class A doesn’t contain main() but static initializers but class A is used in class B containing main() method . Will it execute properly?**  Yes. It will execute properly. As on running class B, it will look for main() method in B, which is available. When class A is required for the first time it will load it in the  memory, static initializers will be executed only first time on next usage previously loaded class itself will be used. | |
| 24 | | **How many times does main() method get executed by java command?**  java command executes main() method only once. | |
| 25 | | **Can main() be executed more than once from a java program?**  Yes, main() method can be executed more than once. | |
| 26 | | **Can a main() method be called from static initializers? How many times?**  Yes main() method can be called from static initializer any no. of times. | |
| 27 | **Is it possible to supply a null while calling a main() method?**  Yes, null can be supplied in a main() method call. | |
| 28 | **Is it possible to have a java file with 2 classes each containing main() method?**  **Which main() method will be executed?**  Yes. In a same java file 2 classes can contain main() method. A classname which is specified for execution that class’s main () method will be executed. | |