**FACILITATOR’S MANUAL**

|  |
| --- |
| Facilitator’s manual is a guideline to facilitator. Guideline for which all topics /sub-topics to be covered and their sequence. When to go recap or hands-on and with which assignment (mapping of lab assignments with topics)  Basically WHAT–WHEN-HOW  Here, Whole session will be in multiple iteration of 3 steps;  1. What to facilitate, 2. Relevant LAB assignments, 3. Recap and leanings from LAB  Also, there are TIPS (extract from facilitator’s learning) – objective of TIPS is to incorporate best practice and individual’s innovation in facilitating a particular topic. It is desirable that new tips should continue to add/update in this manual.  At last, this is not a rulebook, so it is upto facilitator to follow it or use his/her own style |

**Topic – Interface and Packages**

**Objective -**  To understand abstract classes, interfaces, implementation of interfaces, inheritance in interfaces, packages and access modifiers / visibility criteria in Java.

**ROUND 1**

|  |  |
| --- | --- |
| Topics to be facilitated (teach) | * Abstract class |
| LAB assignment | **LAB 4.1**  *Write an abstract class; Try following variations*   * *create an abstract class without keeping any method abstract* * *Make some methods abstract, but do not use abstract word (other modifier) in class’s signature* * *Try to create object of the abstract class, and call abstract method* |
| Recap (learning from the LAB assignment) | At least one method of an abstract class must be abstract.  Abstract is mandatory modifier for method and class to be abstract  Object of abstract class cannot be created. |

**ROUND 2**

|  |  |
| --- | --- |
| Topics to be facilitated (teach) | * Interface * Implementation of an interface * Difference between abstract class and interface |
| LAB assignment | **LAB 4.2**  *Write an interface(IMyMath) , then write a class MyMath, which should implements this interface. In the MainClass (class having main method) create reference (instance) of the interface.*  **LAB 4.3**  *Now write some additional methods in MyMath class (which is implementing IMyMath), i.e. apart from method in interface add some more methods in class. Now in MainClass (main method) create reference instance of interface and instance of MyMath class. Check which instance have which all methods.*  *Similarly, write one more class implanting IMyMath, so now one interface is being implementing by two different classes; write different implementation (body) in both the classes. Create reference instance of using both the classes; now check which instance is accessing which method.* |
| Recap (learning from the LAB assignment) | Even if a class is implementing an interface; the reference instance of the interface will have access of methods in interface only and not to extra/other methods of the class.  The implementation/body of method will be considered for that class, whose constructor is being used to create reference instance of interface. |

**ROUND 3**

|  |  |
| --- | --- |
| Topics to be facilitated (teach) | * Interface extends |
| LAB assignment | **LAB 4.4**  *Write an interface which extends another interface (this exercise is to check inheritance in interfaces)*  *Now write a class implementing the sub-interface and check weather this class have to provide implantation to all the methods of sub and super interface.*  **LAB 4.5**  *Write two or more interface and write a class implementing all interfaces (one or more than one interfaces)*  *Try to extends more than one class* |
| Recap (learning from the LAB assignment) | An Interface can extends another interface  A class can implements many interfaces but can extends only one class. |

**ROUND 4**

|  |  |
| --- | --- |
| Topics to be facilitated (teach) | * Packages * Benefit of packages * Visibility criteria / access modifier * Proper nomenclature for package |
| LAB assignment | **LAB 4.6**  *Create two packages and write few classes and subclasses in packages; now check the various access conditions and find out behavior of various access modifier of method* |
| Recap (learning from the LAB assignment) | What is package, standard nomenclature for packages  Benefit of packages   * Easy to manage * Same class name can be in different package * easy to achieve security/visibility conditions   First line of any class will be package, then import  Below is matrix of access modifier restriction/visibility   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Method access modifier | Within same package | In diff package | Sub class in same package | Sub class in different package | | Public | Y | Y | Y | Y | | Protected | Y | N | Y | Y | | Default | Y | N | Y | N | | Private | N | N | N | N | |

**ROUND 5**

|  |  |
| --- | --- |
| Topics to be facilitated (teach) | * Recap of all modifier * Modifier – Native, transient, volatile |
| LAB assignment | **LAB 4.7**  *Create a matrix of all modifiers and its applicability with various things in Java*  **X – axis of matrix will be various modifiers**  Public  Protected  Private  Default  Static  Final  Abstract  Native  Transient  Volatile  ….  **Y-axis will be**  Class  Method  Variable  Interface  Inner class  Local class  Constructor  ……. |
| Recap (learning from the LAB assignment) | Understanding of all modifiers and which one is applicable with what |