Week 3- Session 2

1. What is an interface in Java?

An interface in Java is a mechanism that is used to achieve complete abstraction. It is basically a kind of class that contains only constants and abstract methods.

2. **Can we define private and protected modifiers for data members (fields) in interfaces?**

No, we cannot define private and protected modifiers for variables in interface because the fields (data members) declared in an interface are by default public, static, and final.

3. Which modifiers are allowed for methods in an Interface?

Only abstract and public modifiers are allowed for methods in interfaces.

4. Can we declare an interface as final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

5. **Can we create an object for an interface?**

- No, we cannot create an object of an interface

6**. Can a method inside an interface be declared as final?**

- By default, methods declared inside an interface are public & abstract even if we don't declare it explicitly compiler adds this modifier during compilation time

- An interface allows only public & abstract modifiers in a method declaration
- If final keyword added in method declaration then the compiler will throw an error as seen in the below screen capture
- Compile-time error: Illegal modifier for the interface method display; only public & abstract are permitted

7**. What happens if we don't initialize variables inside Interface?**

- Compiler throws an error stating final variable needs to be initialized
- As variables defined inside an interface are by default public, static & final. So, the final variable always needs to be initialized where it is declared

8. Can abstract class implements interface?

Yes, an abstract class can implement an interface and this is allowed

9. Can an abstract class be defined without any abstract methods?

- Yes, a class can be declared with abstract keyword even if it doesn't get 1 abstract method
- But vice-versa is not true; means if a class contains abstract methods then the class has to be declared with abstract keyword

10. Whether it is mandatory to have abstract methods in an abstract class? If not, why such a design is required?

- It's not mandatory to have abstract methods in an abstract class
- Even without a single abstract method in a class can be declared as abstract
- This is to flag compiler that this class is not for instantiation

11. Can we define an abstract class without an abstract method? Why it is needed?

- Yes, a class can be declared with abstract keyword even if it doesn't get 1 abstract method
- This is to flag compiler that this class is not allowed to instantiate

12**. Can we define an abstract class without an abstract keyword in the class declaration?**

- No, an abstract keyword is required at class declaration to declare an abstract class

13. Can we define constructor inside an abstract class?

- Yes, we can define constructor inside an abstract class
- Both default & parameterized constructors are allowed inside an abstract class

14. Can an abstract class be instantiated?

- No, an abstract class cannot be instantiated
- Instantiating abstract class throws a compile-time error
- Compiler-time error: Cannot instantiate the type <Abstarct_Class_Name>

15. Can an abstract class be final?

- No, an abstract class cannot be final
- Abstract methods need to be implemented; therefore it is overridden in the subclass
- But by marking final, we are restricting it to override

- A compile-time error will be thrown: The abstract method display in type <abstract-class-name> can only set a visibility modifier, one of public or protected