Smart Programming : YouTube Channel

An investment in Knowledge pays the best interest….

Java 8 Features

=> Java Version Features :-

➔ JDK 1.0 & 1.1 Version :-

- Java was created
- AWT event model
- Inner classes
- JavaBeans
- JDBC
- Reflection
- RMI
- JIT compiler for windows platform

➔ JDK 1.2 Version :-

- Collection frameworks
- strictfp keyword
- Swing API
- Java Plugins

➔ JDK 1.3 Version :-

- HotStop JVM
- JNDI (Java Naming & DIrectory Interface)

- JavaSound

➔ **JDK 1.4 Version :-**

- RegularExpression (RegEx)
- Exception Chaining
- NIO
- Logging API
- Image IO API

➔ **JDK 1.5 Version :-**

- Generics
- Annotations
- Autoboxing/Unboxing
- Enumerations
- Varagrs
- Enhanced for-each loop
- New concurrent utilities in java.util.concurrent package

➔ Java SE 6 Version :-

- JDBC 4.0
- Java Compiler API
- Support for Scripting Langauges
- Performance updates
- New GC algorithms

➔ Java SE 7 Version :-

- Strings in switch
- JVM supports the dynamic languages
- Compressed 64-bit pointers

➔ Java SE 8 Version :-

- Default Methods in Interface
- Static Methods in Interface
- Functional Interfaces
- Lambda Expressions
- Pre-defined Functional Interfaces (Predicate, Function, Consumer, Supplier etc)
- Stream API

- Method Reference
- Constructor Reference
- Date-Time API

=> Why Java SE 8 version is important :-

1. It simplifies the java programming

2. Lambda expressions inherits the features/benifits of "functional programming languages" (But still java is OOP language)

3. Java supports parallel programming

4. Today we have only multiple core processors and java also starts utilizing mutiple core functionalities

---

=> Default Methods in Interface :-

➔ Interface can contain only abstract methods which does not have any implementation part and its implementations where provided in the implemented class

➔ But from 1.8 version, we can create default methods having implementation part in interface. And if we are not satisfied with default method implementation, then we can override that method and provide its our new implementation.

➔ Default methods are created by using "default" keyword

➔ Note :

1. We can override default methods
2. We can provide public access-modifier to default methods

➔ Use of default methods :-

1. If we change the interface then it will affect all the implemented classes, but by creating default methods, implemented classes will not be affected.
2. By creating default methods we can provide same implementation for multiple classes in the interface which in turn reduces the number of lines of code

---

=> Static Methods in Interface :-

➔ Till 1.7 version we were not able to create any static method in an interface but from 1.8 version we can create static methods in an interface

➔ static methods have implementation part in the interface only

➔ Static methods are created by using "static" keyword

➔ **Note :**

1. We can use public access-modifier with static method
2. We cannot use default keyword with static method
3. We were not able to override static methods. If we try to override it, it will be treated as different method

➔ Static methods are used to improve the shareability in interface

=> Marker Interface :-

➔ If any interface does not contain any abstract method, then it is known as marker interface

➔ Predefined marker interfaces are Cloneable, Serializable, Remote etc

=> Functional Interfaces :-

➔ If any interface contains only and only one abstract method, then it is known as Functional Interface

➔ To denote function interface we use "@FunctionalInterface" annotation

➔ Cases for Functional Interface :-

1. We can create any number of "default" or "static" methods in functional interface but we can create only and only one asbtract method
2. In case of inheritance if any interface have more than one abstract method, then also it will not be treated as Functional Interface

➔ Predefined functional interfaces are :- Comparable, ActionListener, Runnable, Callable etc

=> Annotations :-

➔ Annotations are used to provide some additional information about the program
➔ Annotations starts with "@"
➔ If we use or dont use annotations then there is no effect on the compiler
➔ Annotations provides some metadata for classes or interfaces or methods or variables or constructors

Company Links & Contacts

Company Name: Smart Programming (+91 62838-30308)

Address : Chandigarh & Mohali (Punjab), India

Websites: https://www.smartprogramming.in/

https://courses.smartprogramming.in

Android App:
https://play.google.com/store/apps/details?id=com.smartprogramming

YouTube Channel:
https://www.youtube.com/c/SmartProgramming