# Java 8 Features

⇒ Java 8 features :-
- -> Default methods in interfaces
- -> Static methods in interfaces
- -> Functional Interfaces
- -> Lambda Expressions
- -> Pre-defined Functional interfaces
  - Predicate, Function, Consumer, Supplier
- -> Stream API
- -> Method Reference
- -> Constructor Reference
- -> Date-Time API

---

⇒ Default Methods in Interfaces :-
- -> By default interfaces contains public abstract methods which does not have any implementation
- -> But from java 8 version, we can create implemented methods (with body)
- -> Default methods are the methods which are declared as default with implementation
- -> Use :
  1. If we change the interface then we have to change the classes or override the methods, so if we create default methods, then there is no need to change the class
  2. If multiple classes has same implementation, then we can provide the common implementation in the interface



⇒ Static methods in interfaces:-
- -> Static methods are the methods which have implementation in the interface
- -> Static methods in interface are used to improve shareability
- -> Program

⇒ Marker Interface :-
- -> If an interface does not have any abstract method, then it is known as marker interface
- -> They acts as tags or information
- -> Syntax :-

```
interface I1 {}
```

- -> Pre-defined marker interfaces :-
  - Serializable
  - Cloneable
  - Remote
    etc

⇒ Functional Interfaces :-
- -> If any interface has only one abstract method, then it is known as functional interface
- -> We denote functional interfaces using @FunctionalInterface annotation
- -> Syntax :-

```
interface i1
{
    // only 1 abstract method
}
```

- -> Pre-defined functional interfaces :-
  - Runnable
  - Comparable
  - Callable
  - ActionListener
    etc

⇒ Lambda Expressions :-
- -> It is a function which does not have :-
  - any name
  - any modifier
  - return type

```
public void m1()
{
    // body
}
```

→

```
public void m1()
{
    // body
}
```

```
() ->
{
    // body
}
```

Lambda Expression

- -> Advantages :-
  1. It reduces the number of lines of code
  2. It is very effective in API calling
  3. It supports parallel and sequential execution
  4. It is used to write readable, maintanable and concise code

- -> NOTE : It is always used with "functional interfaces"

- -> Rules :-
  1. If we have only one statement in lambda expression, then we can remove curly braces {}
  2. Lambda expression can take any number of parameters
  3. Providing data type with parameter in lambda expression is optional.
  4. If we have only one parameter then we can remove round braces.
  5. If lambda expression is returning any value, then we can remove return keyword

# Programs Task

⇒ Easy
1. WAP to find the length of a name using lambda expression.
2. WAP to check if a number is even or odd using lambda expression.
3. WAP to print a string in uppercase using lambda expression.
4. WAP to calculate the square of a number using lambda expression.
5. WAP to print the last character of a string using lambda expression.



⇒ Medium
6. WAP to check if a string is palindrome using lambda expression.
7. WAP to return maximum of two numbers using lambda expression.
8. WAP to return minimum of two numbers using lambda expression.
9. WAP to reverse a string using lambda expression.
10. WAP to check if a given number is prime using lambda expression.

⇒ Tricky
11. WAP to count total vowels in a string using lambda expression.
12. WAP to check if a number is Armstrong number using lambda expression.
13. WAP to find factorial of a number using lambda expression.
14. WAP to implement a simple calculator (+, -, *, /) using lambda expressions.
15. WAP to check if two strings are anagrams using lambda expression.

# Interview Questions

⇒ Default Methods in Interfaces
1. What are default methods in Java interfaces?
2. Why were default methods introduced in Java 8?
3. How do you declare a default method in an interface?
4. Can an interface have both abstract methods and default methods?
5. What happens if two interfaces provide the same default method and a class implements both?
6. Can default methods be overridden in implementing classes?
7. Can constructors be declared as default methods in interfaces? Why or why not?
8. Are default methods inherited by sub-interfaces?

⇒ Static Methods in Interfaces
9. What are static methods in interfaces?
10. How are static methods in interfaces different from default methods?
11. How do you call a static method defined inside an interface?
12. Can static methods in interfaces be inherited by implementing classes?
13. Can you override static methods from an interface in an implementing class? Why or why not?
14. Why were static methods allowed in interfaces starting from Java 8?
15. Can an interface have both static and default methods together?

⇒ Functional Interfaces
16. What is a functional interface in Java?
17. Give some examples of built-in functional interfaces in Java.
18. How do you create a custom functional interface?
19. What is the purpose of the @FunctionalInterface annotation? Is it mandatory?
20. Can a functional interface have multiple abstract methods? Why or why not?
21. Can a functional interface have default and static methods?
22. Is Runnable a functional interface? Why?
23. What will happen if we add another abstract method to a functional interface annotated with @FunctionalInterface?

⇒ Lambda Expressions
24. What is a lambda expression in Java?
25. How are lambda expressions related to functional interfaces?
26. What is the syntax of a lambda expression in Java?
27. Can a lambda expression have multiple statements? How?
28. How does type inference work with lambda expressions?
29. Can lambda expressions access variables from the enclosing scope? If yes, what are the restrictions?
30. What are the advantages of using lambda expressions?
31. Can lambda expressions throw exceptions?
32. What is the difference between an anonymous inner class and a lambda expression?