# Java For Life

## Generics

* Angle braces are used for Generics
* For using same class or same method to support different types in parameters
* Type parameter should not be primitive type, <Integer> instead of <int>
* You can declare a bound parameter just by extending the required class with the type-parameter, within the angular braces as −

**class Sample <T extends Number>**

## Collection

* Collection does not work with index number. List does.

## Functional Interface

## Lambda Expression

**Why Lambdas**

* **Enables** functional programming
* Readable and concise code
* Easier-to-use APIs and libraries
* Enables support for parallel processing
* There must be a functional interface to implement lambda expression
* Lambda expression helps us in achieving functional programming, where we implement the interface by just implementing the function and not implementing the class.

## What/Why Functional Programming

In OOP every thing has to be a part of a class, either it is a property or its is a function. It is sometimes a problem when you want a single piece of work/logic to be alone not a part of a class. You cannot do this in OOP. Here comes the Functional programming.

In OOP you always need a class or you write your logic in an interface that is eventually implemented by a class. So, there is always a class. To remove class, you use functional programming.

## Functional Interface

* It has only one abstract method, but one or more implemented methods
* In java 8, you can have interfaces that can have concrete methods,
* **@FunctionalInterface** annotation is optional but you use it to avoid other to add more abstract methods

## Collection Interface

