***Streams***

***What is stream ?  
 It is a sequence of elements supporting functional and declarative programing***

* *Feature introduced in java 8*
* *Process collection in functional and declarative process*
* *Simplify data processing*
* *Embrace Functional Programming*
* *Improve readability and functionality*
* *Enable Easy Parallelism*

*It has* ***Source,Intermediate operations & terminal operations***

***Example***

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
List<String> res = names.stream()  
 .filter(name -> name.startsWith("A"))  
 .collect(Collectors.toList());  
System.out.println(res);

***Source :*** *List<String> names = Arrays.asList("Alice", "Bob", "Charlie");*

***Conversion to stream:*** *names.stream()*

***Intermediate operations:***  *.filter(name -> name.startsWith("A"))*

***terminal operations:*** *.collect(Collectors.toList());*

***Source(Generating of streams)***

1. *Generating from colletions*

List<Integer> list = Arrays.*asList*(1, 2, 3, 4, 5);  
Stream<Integer> stream = list.stream();

1. *Generating from Arrays*

String[] array = {"a", "b", "c"};  
Stream<String> stream1 = Arrays.*stream*(array);

1. *Using Stream of*

Stream<String> stream2 = Stream.*of*("a", "b");

1. *Generating infinite Stream*

Stream.*generate*(() -> 1);  
Stream.*iterate*(1, x -> x + 1).skip(10).limit(2000);