Traversing Dynamic Collection with Lambda

Dynamic collections can use the normal for loop to traverse through the elements or they can use special functions call Lambda. Lambda is a method which can be passed as parameter.

```
aList.forEach(arrElement-> {
System.out.printf("\t\t%s\n", arrElement);
});
```

In the above code, aList refers to an ArrayList (or any other valid collection object). 'forEach' takes a Consumer as a parameter. Consumer is a method which will take one parameter and do something with it and return nothing. In this case, we are extracting each element in the aList as arrElement and format printing them within the lambda block.

Collections can also be filtered using lambda. Think of an ArrayList which contains all numbers from 1 to 100. The following code will filter out only the odd numbers. The filter lambda takes a Predicate. Predicate is a method which takes one argument and returns a boolean based on how it processes/check the argument.

```
        ArrayList naturalNumbers = new ArrayList<>();

        for(int i=0;i<100;i++) {
            naturalNumbers.add(i);
        }

        naturalNumbers.stream().filter(num->{
            return num%2 == 1;
        }).collect(Collectors.toList());
```

This picture depicts how the filter works.

Screen Shot 2019-08-13 at 7.52.51 pm.png