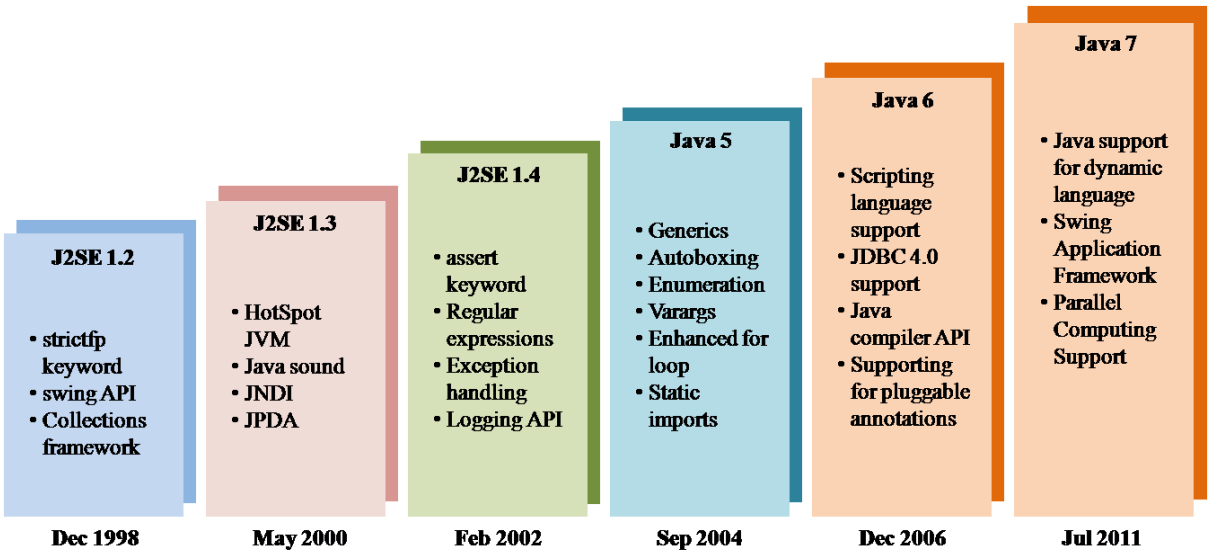
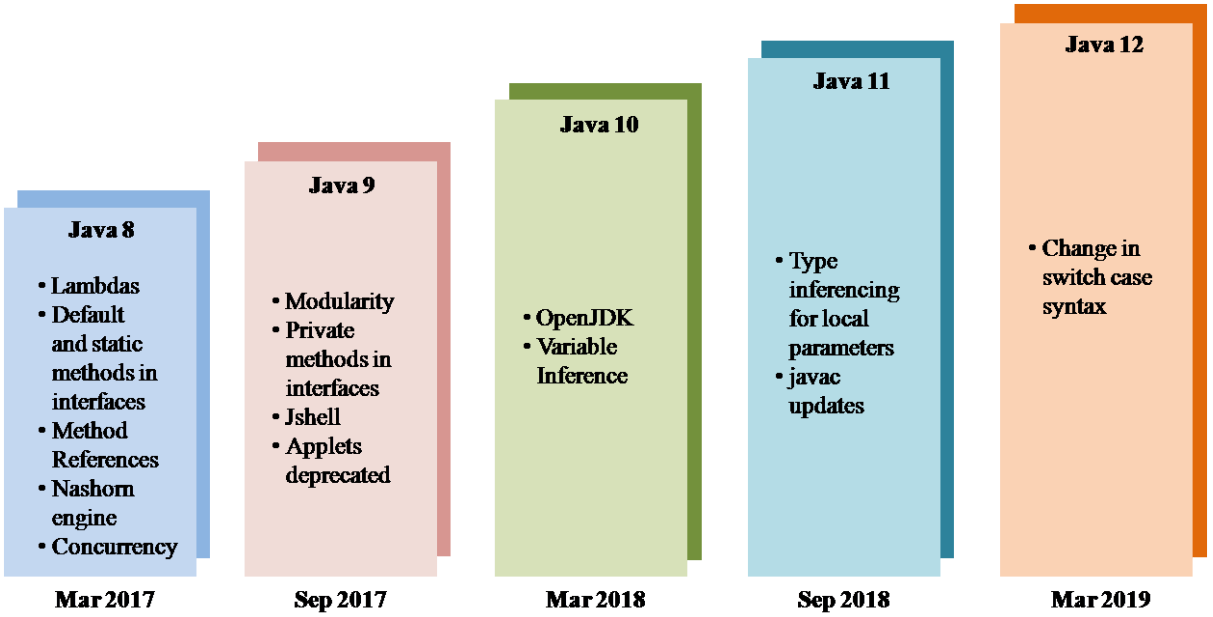
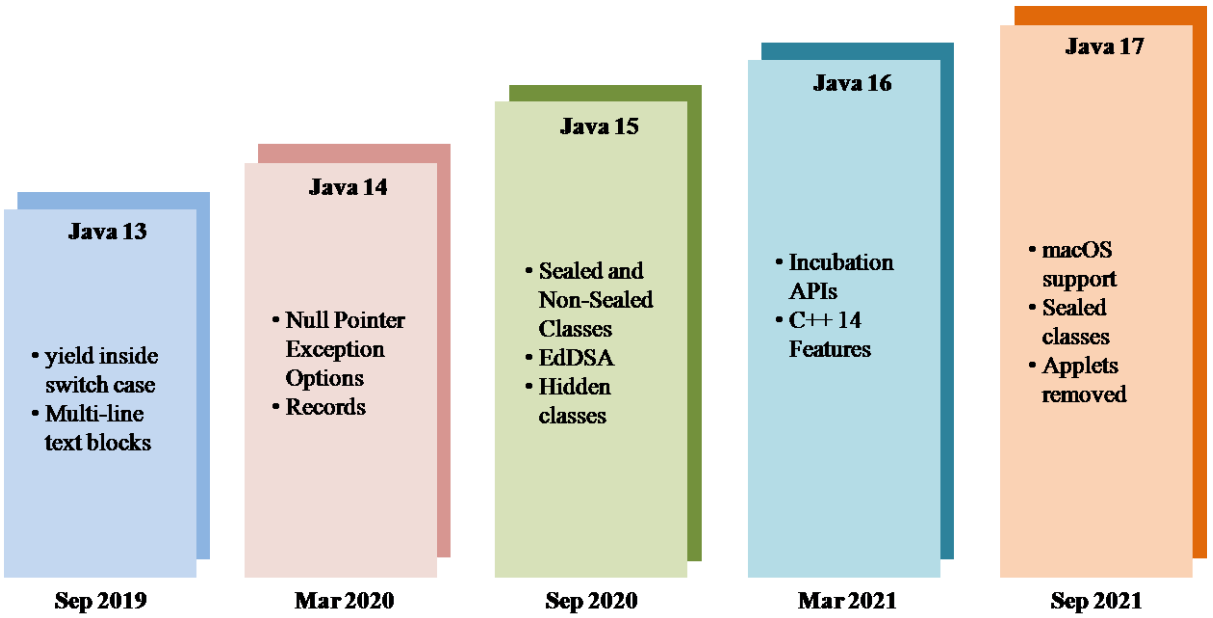
Java New Features







- Lambda Expressions
- Functional Interfaces
- Default Methods
- Predicates
- Functions
- Stream API
- Date API

Few examples of functional programming:

```
public class StructuralProgramming {
    public static void main(String[] args) {
        List<Integer> numbers = List.of(12, 9, 13, 4, 6, 2, 4, 12, 15);
        System.out.println("Printing all numbers in list");
        for (Integer number : numbers) {
            System.out.println(number);
        }
        System.out.println("Printing all even numbers in list");
        for (Integer number : numbers) {
            if (number % 2 == 0) {
                System.out.println(number);
            }
        }
    }
}
```

Output:

```
Printing all numbers in list
12
9
13
4
6
2
4
12
15
```

```
Printing all even numbers in list
12
4
6
2
4
12
```

```java
public class StructuralProgrammingRefactored {
    public static void main(String[] args) {
        List<Integer> numbers = List.of(12, 9, 13, 4, 6, 2, 4, 12, 15);
        printAllNumbersInList(numbers);
        printAllEvenNumbersInList(numbers);
    }

    private static void printAllEvenNumbersInList(List<Integer> numbers) {
        System.out.println("Printing all even numbers in list");
        for (Integer number : numbers) {
            if (number % 2 == 0) {
                System.out.println(number);
            }
        }
    }

    private static void printAllNumbersInList(List<Integer> numbers) {
        System.out.println("Printing all numbers in list");
        for (Integer number : numbers) {
            System.out.println(number);
        }
    }
}
```

Output:
Same.

```java
public class FunctionalProgrammingRefactored {
    public static void main(String[] args) {
        List<Integer> numbers = List.of(12, 9, 13, 4, 6, 2, 4, 12, 15);
        printAllNumbersInList(numbers);
        printAllEvenNumbersInList(numbers);
        printAllOddNumbersInList(numbers);
    }

    private static void printAllEvenNumbersInList(List<Integer> numbers) {
        System.out.println("Printing all even numbers in list");
        numbers.stream()
                .filter(i -> i % 2 == 0)
                .forEach(System.out::println);
    }

    private static void printAllOddNumbersInList(List<Integer> numbers) {
        System.out.println("Printing all odd numbers in list");
        numbers.stream()
                .filter(i->i%2==1)
                .forEach(System.out::println);
    }
```

```
    private static void printAllNumbersInList(List<Integer> numbers) {
        System.out.println("Printing all numbers in list");
        numbers.stream().
                forEach(System.out::println);
    }
}
```

Output

```
Printing all numbers in list
12
9
13
4
6
2
4
12
15
Printing all even numbers in list
12
4
6
2
4
12
Printing all odd numbers in list
9
13
15
```

```
public class FunctionalProgrammingRefactoredStringExamples {
    public static void main(String[] args) {
        List<String> javaTopics = List.of("Exception Handling", "Files",
                    "Threads", "Collections", "JVM", "Garbage Collection", "OOPS");
        printAllJavaTopics(javaTopics);
        printLengthOfEachTopic(javaTopics);
        printSingleWordTopicsWithGivenLength(javaTopics);
    }

    private static void printSingleWordTopicsWithGivenLength(List<String> javaTopics) {
        System.out.println("Printing single word topic with length greater than 5");
        javaTopics.stream()
                .filter(s -> s.length() > 5)
                .filter(s -> !s.contains(" "))
                .map(s -> s + " " + s.length())
                .forEach(System.out::println);
    }

    private static void printLengthOfEachTopic(List<String> javaTopics) {
        System.out.println("Printing length of each java topic");
        javaTopics.stream()
                .map(s -> s + " " + s.length())
                .forEach(System.out::println);
    }

    private static void printAllJavaTopics(List<String> javaTopics) {
        System.out.println("Printing all java topics");
        javaTopics.stream()
                .forEach(System.out::println);
    }
}
```

Printing all java topics
Exception Handling
Files
Threads
Collections
JVM
Garbage Collection
OOPS
Printing length of each java topic
Exception Handling 18
Files 5
Threads 7
Collections 11
JVM 3
Garbage Collection 18
OOPS 4
Printing single word topic with length greater than 5
Threads 7
Collections 11