Local Variable Type Inference:

Java added a new way to declare and initialize Java variables by using var keyword. It reduces the effort to specify the type during variable declaration. By using var, we don't need to use types such as int, float, etc. It automatically infers the type of the variable based on the assigned value. This feature is known as Local Variable Type Inference or LVTI in Java.

Code:

```
public class LocalVariableTypeInference {
 public static void main(String[] args) {

 int a = 2344324;
 var b = 2344324;

 String str = "Value";
 var str2 = "Value";

 System.out.println(a == b);
 System.out.println(str.equals(str2));
 }
}
```

Output:

```
true

true
```

Optional*.orElseThrow():

java.util.Optional, java.util.OptionalDouble, java.util.OptionalIntand java.util.OptionalLongeach got a new method orElseThrow()which doesn’t take any argument and throws NoSuchElementExceptionif no value is present

Code:

```
import java.util.Optional;

public class OptionalDemo {

 public static void main(String[] args) {
```

```
 String[] arr = new String[10];
 Optional<String> op = Optional.ofNullable(arr[5]);
 op.ifPresent(System.out::println);
 op.orElseThrow(NullPointerException::new);
 }
}
```

Output:

```
Exception in thread "main" java.lang.NullPointerException
at java.base/java.util.Optional.orElseThrow(Optional.java:403)
at OptionalDemo.main(OptionalDemo.java:9)
```

Unmodifiable Collections:

a.) copyOf()

java.util.List, java.util.Map and java.util.Set each got a new static method copyOf(Collection).

b.) toUnmodifiable*()

java.util.stream.Collectors get additional methods to collect a Stream into unmodifiable List, Map or Set

Code:

```
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class UnModDemo {

 public static void main(String[] args) {
 List<Integer> list = Arrays.asList(1, 2, 3, 4, 6, 7, 9);
 Set<Integer> set = Set.of(1, 2, 3, 4);
 Map<String, Integer> map = Map.of("a", 1, "b", 2, "c", 3);
 Stream<Integer> s1 = Stream.of(1, 2, 3, 4);
 Stream<String> s2 = Stream.of("rover", "range", "starting", "hunter");

 List<Integer> lis = s1.collect(Collectors.toUnmodifiableList());
 Set<String> se = s2.collect(Collectors.toUnmodifiableSet());
 List<Integer> obj1 = List.copyOf(list);
 obj1.add(234);

 Set<Integer> set1 = Set.copyOf(set);
 set1.add(535);

 System.out.println(set1);
 System.out.println(obj1);
 }
}
```

Output:

```
Exception in thread "main" java.lang.UnsupportedOperationException

at java.base/java.util.ImmutableCollections.uoe(ImmutableCollections.java:142)

at
java.base/java.util.ImmutableCollections$AbstractImmutableCollection.add(ImmutableCollectio
ns.java:147)

at UnModDemo.main(UnModDemo.java:21)
```