Explanation A

The method *convert(src, toType)* converts an object *src* to a specified type *toType*. The assertion in line 4 checks if the method outputs the same value as the input variable *src*.

Internally, the method computes and returns the value of the variable *result* based on its *src* parameter,
However,
1. In V1, the method first checks if *src* is not *null* and is assignable to the target type *toType*. Since this condition evaluates to *true*, *src* is returned directly.
2. In V2, this condition check and the direct return of *src* have been removed. Instead, *result* is computed as a function of *src* and then returned.

In summary, removing code that returns the value of the input *src* variable directly (line 8 in V1) and rather returning a computed value (line 14 in V2) causes the returned value in V2 to differ from the value of the input variable *src*, leading to the assertion failure in line 4.

Explanation B

The method *convert(src, toType)* converts an object *src* to a specified type *toType*. The assertion in line 4 checks if the method outputs the same value as the input variable *src*.

Internally, the method computes and returns the value of the variable *result* based on its *src* parameter,
However,
1. In V1, the method first checks if *src* is not *null* and is assignable to the target type *toType*. Since this condition evaluates to *true*, *src* is returned directly.
2. In V2, this condition check and the direct return of *src* have been removed. Instead, a new *TokenBuffer* is created, and *src* is serialized into this buffer. The buffer is then used to create a *JsonParser*, which, combined with a configuration object obtained from the serialization step, is used to deserialize *src* into result. Finally, the deserialized object is returned.

In summary, removing code that returns the value of the input *src* variable directly (line 8 in V1) and rather returning a computed value (line 14 in V2) causes the returned value in V2 to differ from the value of the input variable *src*, leading to the assertion failure in line 4.

Explanation C

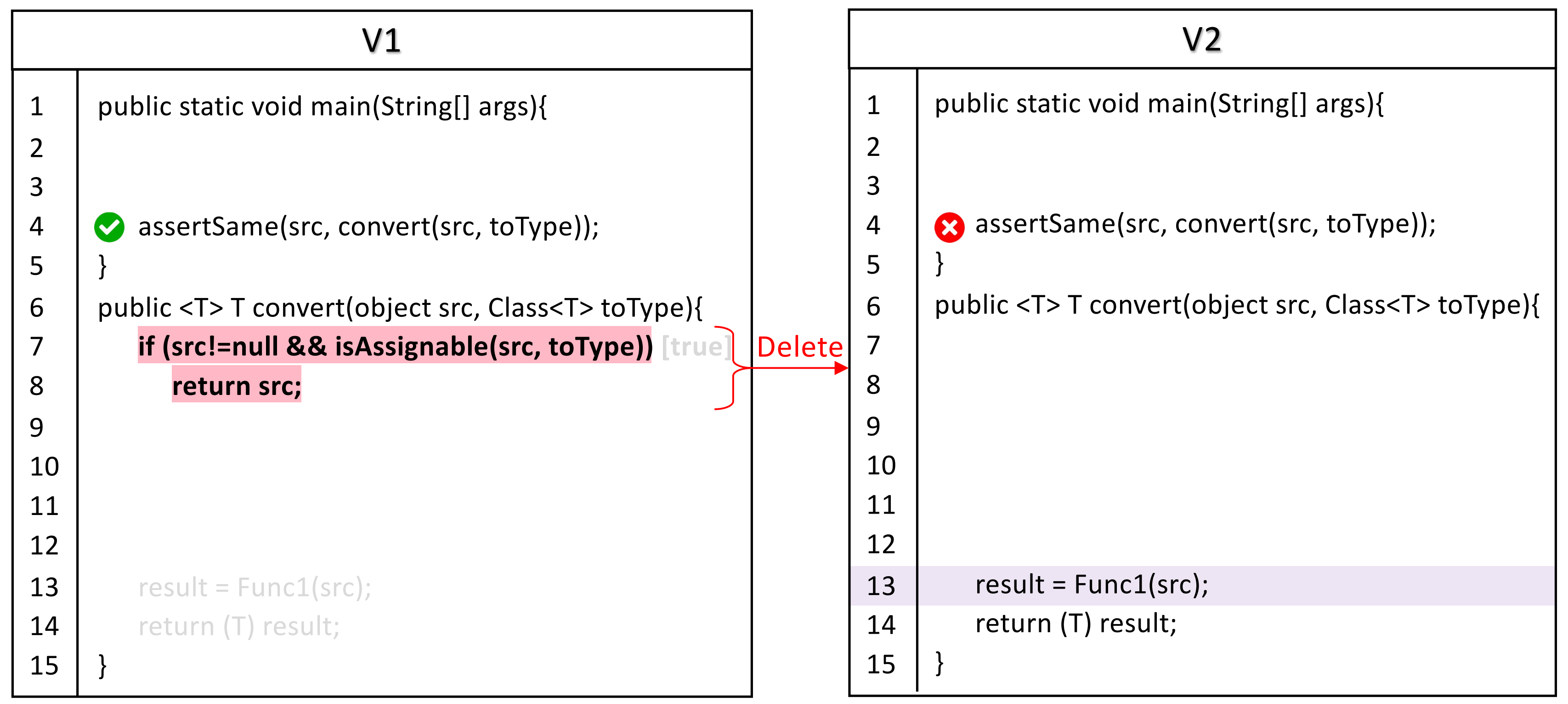
The method *convert(src, toType)* converts an object *src* to a specified type *toType*. The assertion in line 4 checks if the method outputs the same value as the input variable *src*,
which is initialized in line 2 to be an instance of *TestObject*.
The desired conversion type is initialized in line 3 to be *TestObject* as well.
Internally, the method computes and returns the value of the variable *result* based on its *src* parameter,
However,
1. In V1, the method first checks if *src* is not *null* and is assignable to the target type *toType*. Since this condition evaluates to *true*, *src* is returned directly.
2. In V2, this condition check and the direct return of *src* have been removed. Instead, *result* is computed as a function of *src* and then returned.

In summary, removing code that returns the value of the input *src* variable directly (line 8 in V1) and rather returning a computed value (line 14 in V2) causes the returned value in V2 to differ from the value of the input variable *src*, leading to the assertion failure in line 4.

<u>Notations</u>: Colored backgrounds highlight the differences between the views.

FYI: Views are given below again, for your reference.

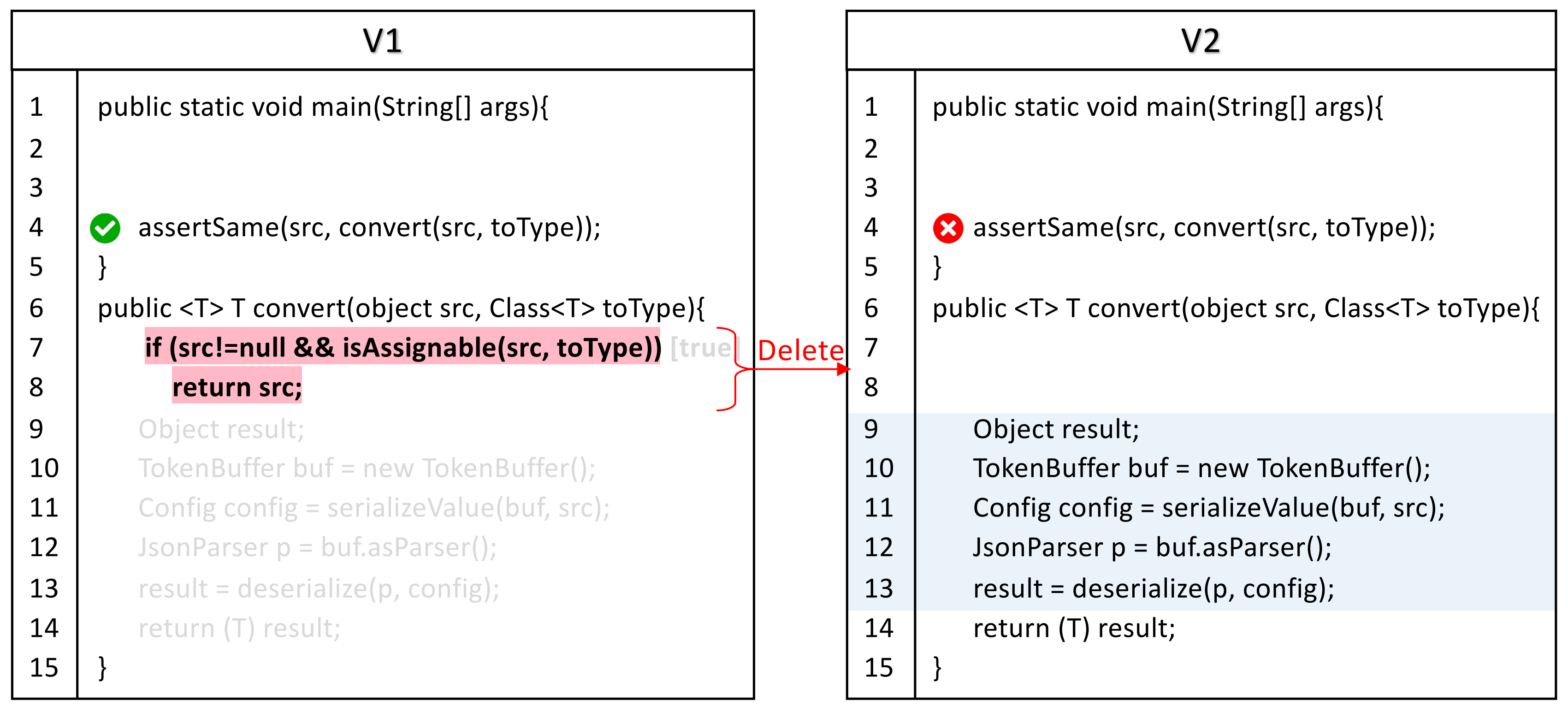
View A



V1
```
1   public static void main(String[] args){
2
3
4     ✓ assertSame(src, convert(src, toType));
5   }
6   public <T> T convert(object src, Class<T> toType){
7       if (src!=null && isAssignable(src, toType))
8          return src;
9
10
11
12
13      result = Func1(src);
14      return (T) result;
15  }
```

V2
```
1   public static void main(String[] args){
2
3
4     ✗ assertSame(src, convert(src, toType));
5   }
6   public <T> T convert(object src, Class<T> toType){
7
8
9
10
11
12
13      result = Func1(src);
14      return (T) result;
15  }
```

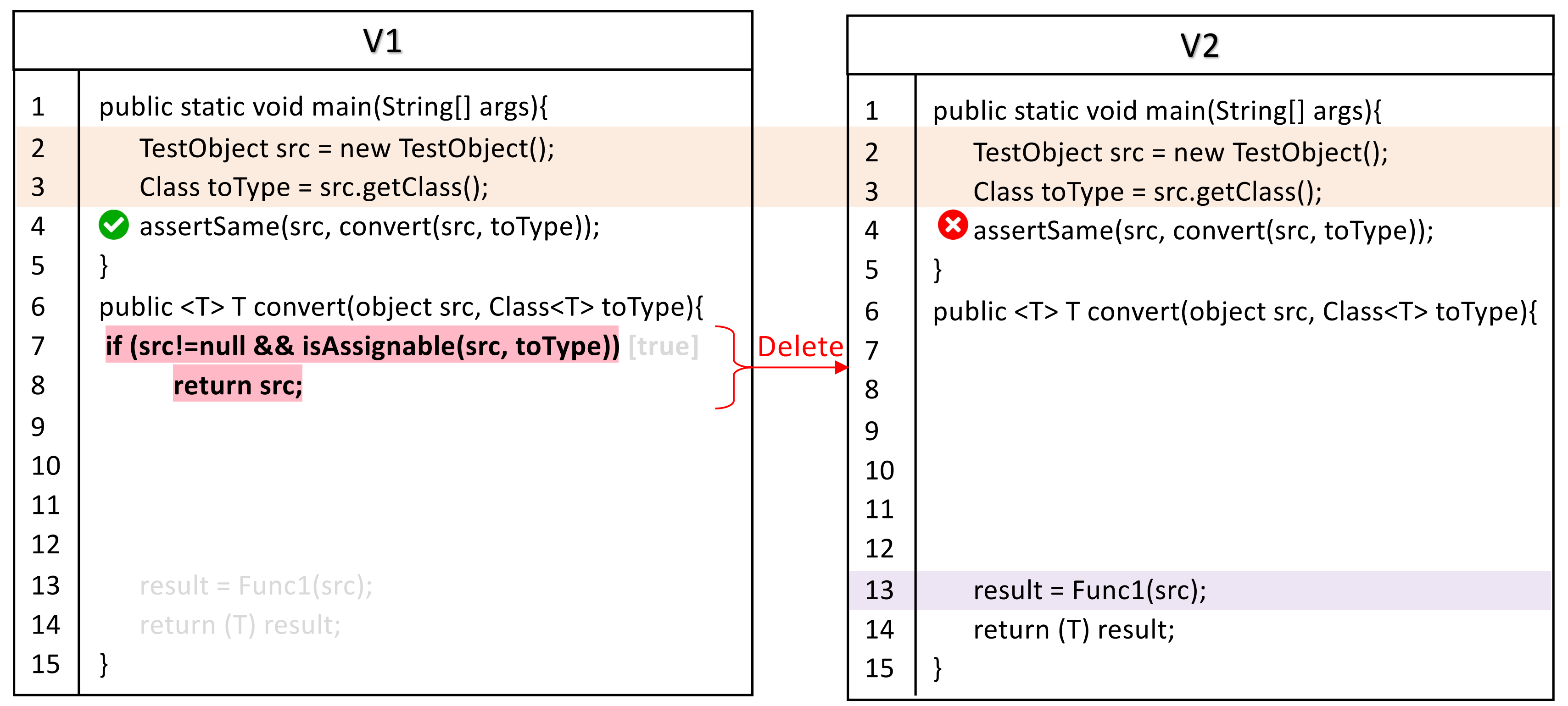
View B



V1
```
1   public static void main(String[] args){
2
3
4     ✓ assertSame(src, convert(src, toType));
5   }
6   public <T> T convert(object src, Class<T> toType){
7       if (src!=null && isAssignable(src, toType))
8          return src;
9       Object result;
10      TokenBuffer buf = new TokenBuffer();
11      Config config = serializeValue(buf, src);
12      JsonParser p = buf.asParser();
13      result = deserialize(p, config);
14      return (T) result;
15  }
```

V2
```
1   public static void main(String[] args){
2
3
4     ✗ assertSame(src, convert(src, toType));
5   }
6   public <T> T convert(object src, Class<T> toType){
7
8
9       Object result;
10      TokenBuffer buf = new TokenBuffer();
11      Config config = serializeValue(buf, src);
12      JsonParser p = buf.asParser();
13      result = deserialize(p, config);
14      return (T) result;
15  }
```

View C



V1
```
1   public static void main(String[] args){
2     TestObject src = new TestObject();
3     Class toType = src.getClass();
4     ✓ assertSame(src, convert(src, toType));
5   }
6   public <T> T convert(object src, Class<T> toType){
7       if (src!=null && isAssignable(src, toType))
8          return src;
9
10
11
12
13      result = Func1(src);
14      return (T) result;
15  }
```

V2
```
1   public static void main(String[] args){
2     TestObject src = new TestObject();
3     Class toType = src.getClass();
4     ✗ assertSame(src, convert(src, toType));
5   }
6   public <T> T convert(object src, Class<T> toType){
7
8
9
10
11
12
13      result = Func1(src);
14      return (T) result;
15  }
```