

View A

V1		V2	
1	public static void main(String[] args){	1	public static void main(String[] args){
2		2	
3		3	
4	✔️ assertSame(src, convert(src, toType));	4	❌ assertSame(src, convert(src, toType));
5	}	5	}
6	public <T> T convert(object src, Class<T> toType){	6	public <T> T convert(object src, Class<T> toType){
7	if (src!=null && isAssignable(src, toType)) [true]	7	
8	return src;	8	
9		9	
10		10	
11		11	
12		12	
13	result = Func1(src);	13	result = Func1(src);
14	return (T) result;	14	return (T) result;
15	}	15	}

View B

V1		V2	
1	public static void main(String[] args){	1	public static void main(String[] args){
2		2	
3		3	
4	✔️ assertSame(src, convert(src, toType));	4	❌ assertSame(src, convert(src, toType));
5	}	5	}
6	public <T> T convert(object src, Class<T> toType){	6	public <T> T convert(object src, Class<T> toType){
7	if (src!=null && isAssignable(src, toType)) [true]	7	
8	return src;	8	
9	Object result;	9	Object result;
10	TokenBuffer buf = new TokenBuffer();	10	TokenBuffer buf = new TokenBuffer();
11	Config config = serializeValue(buf, src);	11	Config config = serializeValue(buf, src);
12	JsonParser p = buf.asParser();	12	JsonParser p = buf.asParser();
13	result = deserialize(p, config);	13	result = deserialize(p, config);
14	return (T) result;	14	return (T) result;
15	}	15	}

View C

V1		V2	
1	public static void main(String[] args){	1	public static void main(String[] args){
2	TestObject src = new TestObject();	2	TestObject src = new TestObject();
3	Class toType = src.getClass();	3	Class toType = src.getClass();
4	✔️ assertSame(src, convert(src, toType));	4	❌ assertSame(src, convert(src, toType));
5	}	5	}
6	public <T> T convert(object src, Class<T> toType){	6	public <T> T convert(object src, Class<T> toType){
7	if (src!=null && isAssignable(src, toType)) [true]	7	
8	return src;	8	
9		9	
10		10	
11		11	
12		12	
13	result = Func1(src);	13	result = Func1(src);
14	return (T) result;	14	return (T) result;
15	}	15	}

Notations: Lines that correspond to each other in V1 and V2 are given the same line numbers. Numbered empty lines indicate statements that are either excluded in a particular view or are absent in a code version. Specifically, if a line is annotated by "delete", the statement is deleted in a version. Otherwise, it is excluded from the view. Changes between versions (“Add”, “Update”, “Delete”) are shown on arrows between code snippets. For simplicity, each “if” statement (e.g., in line 7) is annotated with a label showing whether the “if” condition is evaluated to *true* or *false*. Gray lines indicate statements that are not executed because an 'if' statement (e.g. line 7) that evaluates to *true* prevents their execution. Colored background highlight the differences between the views.