<InterfaceDecl>	::=	[<InterfaceMod>] 'interface' <Identifier>
		['extends' <Identifier> { ',' <Identifier> }]
		(<InterfaceBody> | ';')
<InterfaceMod>	::=	('public' | 'protected' | 'private')
<Identifier>	::=	<Letter> { (<Letter> | <Digit>) }
<InterfaceBody>	::=	'{' { <InterfaceMemD> } '}'
<InterfaceMemD>	::=	('method' | 'constant' | 'class')
<Letter>	::=	('a' | 'b' | '$')
<Digit>	::=	('0' | '1')

(1) Interface ab1a1a1;

Geht, da am Ende eine oder Alternative ist. → Entweder { } oder ;
InterfaceMod lässt man 0, interface wird rüber genommen, in Identifier wird in Letter gegangen, da dann ein a, in {} wird Letter und Digit wiederholt.

(2) Public interface aaa extends bbb, ab$ {method method}

Geht:

InterfaceMod → Public, interface wird übernommen, Identifier → Letter → a → Letter → a 2 mal, extends wird übernommen, Identifier → Letter → b → Letter → b 2 mal, Komma wird übernommen, Identifier → Leter → a → Letter → b → Letter → $, InterfaceBody → {, InterfaceMemD → method → InterfaceMemD → method → }

(3) Protected interface 0$ab extends aaa method constant

Geschweifte Klammer muss InterfaceMemD umschließen. Nicht der Fall, demnach Syntaxfehler.

(4) interface aaa, bbb extends aaa, bbb { aaa }

Das erste Komma kann man nicht holen, demnach Syntaxfehler.

(5) InterfaceMod interface extends Identifier ;

Fängt mit InterfaceMod an, ist nicht möglich, demnach Syntaxfehler.

(6) private public interface extends aaa , { class extends bbb }

InterfaceMod ist in eckigen Klammern, demnach kann private und public nicht nebeneinander stehen.