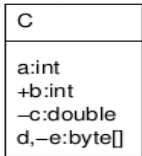
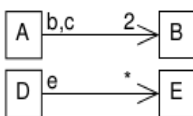
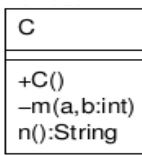

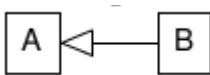
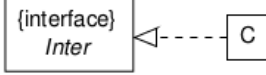
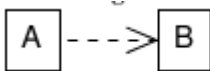
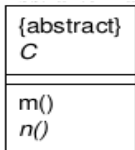
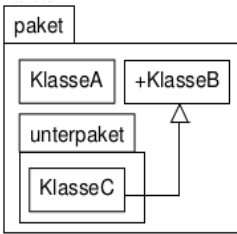
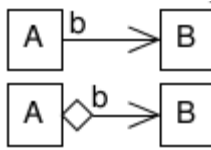
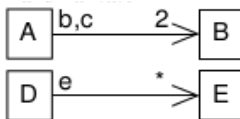


Q: Given are UML-diagrams(1...11) and java code(A...K). Which diagram fits to which java code?

A: Fill in the appropriate letter: 1-__, 2-__, 3-__, 4-__, 5-__, 6-__, 7-__, 8-__, 9-__, 10-__, 11-__

1	2	3	4	5	6
					
7	8	9	10	11	
					

A	B	C	D	E	F
<pre>class A {B b,c;} class D {E e[]; // zum Bsp. }</pre>	<pre>class C { int a; public int b; private double c; ...}</pre>	<pre>class C { public C () {...} private m(int a, int b){...} String n(){...}</pre>	<pre>class C { public static void main()... ...</pre>	vieldeutig: A ruft eine Methode v. B oder A legt eine Instanz von B an	<pre>class C implements Inter {...}</pre>
G	H	I	J	K	
<pre>class B extends A{...}</pre>	<pre>abstract class C { void m() {...} abstract void n() {...} }</pre>	<pre>A.java package paket; class A {...} B.java package paket; class B {...} C.java package paket.unterpaket; class C extends B{...}</pre>	<pre>class A {B b,c;} class D {E e[];}</pre>	In Java ist der Unterschied zw. Assoziation u. Aggregation eher unklar. Auf alle Fälle gilt: die Klasse A hat ein Attribut vom Typ B namens b: class A {B b};	

A: Fill in the appropriate letter: 1-___, 2-___, 3-___, 4-___, 5-___, 6-___, 7-___, 8-___, 9-___, 10-___, 11-___

```
classDiagram
    class IrgendEineKlasse
    class Klasse {
        attributname1 : attributtyp1
        attributname2 : attributtyp2 = defaultwert
        -privatesAttribut : attributtyp
        +publicAttribut : attributtyp
        statischesAttribut : attributtyp
        Klasse()
        operation1(): rueckgabetyyp
        operation2(parametertyp1, parametertyp2)
        operation3(parameternamenam: parametertyp): rueckgabetyyp3
        -privateOperation()
        +publicOperation()
    }
    class Vererbung["Vererbung ('Klasse' erbt von 'IrgendEineKlasse')"]
    class AbstrakteKlasse {
        <<abstract>>
        konkreteMethode()
        abstrakteMethode()
    }
    class NochEineKlasse {
        viele : KlasseKlasse[]
        dumdidum()
    }
    class "1"
    class "eines"
    class "viele"
    class "viele"
    class "KlasseKlasse" {
        methode()
    }
    class Interface {
        <<interface>>
        Interface
        methode()
    }
    class Dings
    class "Dings"

    IrgendEineKlasse <|.. Klasse
    Klasse --> AbstrakteKlasse : benutzt-Beziehung ("AbstrakteKlasse" benutzt "NochEineKlasse")
    AbstrakteKlasse <|-- NochEineKlasse
    Klasse "1" *-- "Dings" : Aggregation: "Klasse" hat ein Attribut namens "dings" vom Typ "Dings"
    Klasse "1" ..> Interface : "KlasseKlasse" implementiert "Interface"
    Interface <|-- KlasseKlasse
    KlasseKlasse "*" *-- "viele" : "NochEineKlasse" hat beliebig viele Attribute vom Typ "KlasseKlasse" im Attribut "viele"
```

The diagram illustrates several UML concepts:

- Inheritance:** A solid arrow points from `Klasse` to `IrgendEineKlasse`. A note states: "Vererbung ('Klasse' erbt von 'IrgendEineKlasse')".
- Abstract Class:** `AbstrakteKlasse` is an abstract class (indicated by {abstract}). It has methods `konkreteMethode()` and `abstrakteMethode()`.
- Generalization:** A solid arrow points from `NochEineKlasse` to `AbstrakteKlasse`, indicating inheritance.
- Association:** A solid arrow connects `Klasse` and `NochEineKlasse`. The multiplicity at `Klasse` is `1` and at `NochEineKlasse` is `eines`. A note explains: "benutzt-Beziehung ('AbstrakteKlasse' benutzt 'NochEineKlasse')".
- Aggregation:** A hollow diamond connects `Klasse` and `Dings`. The multiplicity at `Dings` is `1`. A note states: "Aggregation: 'Klasse' hat ein Attribut namens 'dings' vom Typ 'Dings'".
- Implementation:** A hollow triangle points from `KlasseKlasse` to `Interface`. A note says: "'KlasseKlasse' implementiert 'Interface'".
- Association:** A dashed arrow connects `KlasseKlasse` and `NochEineKlasse`. The multiplicity at `KlasseKlasse` is `*` and at `NochEineKlasse` is `viele`. A note explains: "'NochEineKlasse' hat beliebig viele Attribute vom Typ 'KlasseKlasse' im Attribut 'viele'".

Abbildung 1: Beispiel für ein UML-Klassendiagramm