Modulul Practic - Arhivă

# Descriere Generală

## Descriere Aplicație

Aplicația PGI Archiver permite arhivarea diverselor tipuri de documente in arhive.

O Arhivă poate avea un număr de N documente. Aplicația administrează un număr M de arhive.

Arhivă M

54rt4gt

DocN

Doc1

Arhivă 2

54rt4gt

DocN

Doc1

Arhivă 1

54rt4gt

DocN

Doc1

Numărul documentelor în arhiva este unul variabil.

In mod obisnuit arhiva este administrata de către un administrator. In sistem există și arhive care nu sunt administrate de nimeni.

O arhiva poate fi vizualizată de un anumit număr de utilizatori. Administratorul poate acorda drepturi sau revoca drepturi de acces a utilizatorilor la arhivă. Un utilizator i se poate acorda dreptul de a accesa mai multe arhive.

Arhivă X

54rt4gt

DocN

Doc1

Access

Utilizatori

Administrator

O Arhivă poate avea un număr de N documente. O arhiva poate fi vizualizată de un număr M de utilizatori. In cazul in care utilizatorul nu se regăsește in această listă M de utilizatori, arhiva nu va fi afișată.

## Descrierea bazei de date

### Tabel ”document”

Tabel ”document”

|  |  |  |  |
| --- | --- | --- | --- |
| Nume Coloana | Tip | Descriere |  |
| document\_id | Char (16) | Id-ul documentului | PK |
| document\_name | Char (32) | Nume document |  |
| archived\_date | Date | Data Arhivării |  |
| document\_size | Integer | Dimensiunea in bytes a fisierului |  |
| archive\_id | Char (16) | Id-ul arhivei |  |

Tabel 1

### Tabel ”arhive”

Tabel ”arhive”

|  |  |  |  |
| --- | --- | --- | --- |
| Nume Coloana | Tip | Descriere |  |
| archive\_id | Char (16) | Id-ul Arhivei – primary key | PK |
| archive\_name | Char (32) | Nume document |  |
| creation\_date | Date | Data Arhivării |  |
| admin\_id | Char (16) |  |  |
| licensing\_mode | Integer | Fiecare tip de licentiere are un cod:  1 – per număr de documente  2 – per volum de stocare |  |

Tabel 2

### Tabel ”user\_access”

Tabel ”user\_access” – acces utilizatori

|  |  |  |  |
| --- | --- | --- | --- |
| Nume Coloana | Tip | Descriere |  |
| user\_id | Char (16) | Id-ul utilizatorului ce poate accesa arhiva |  |
| archive\_id | Char (16) | Id-ul Arhivei la care are acces |  |
| privilege\_name | Integer | Fiecărei valori numerice îi este asociat un privilegiu:   * 1 - READ (Citire) * 2 - WRITE (Scriere) * 3 - DELETE (Ștergere) |  |

Tabel 3

### Tabel ”users”

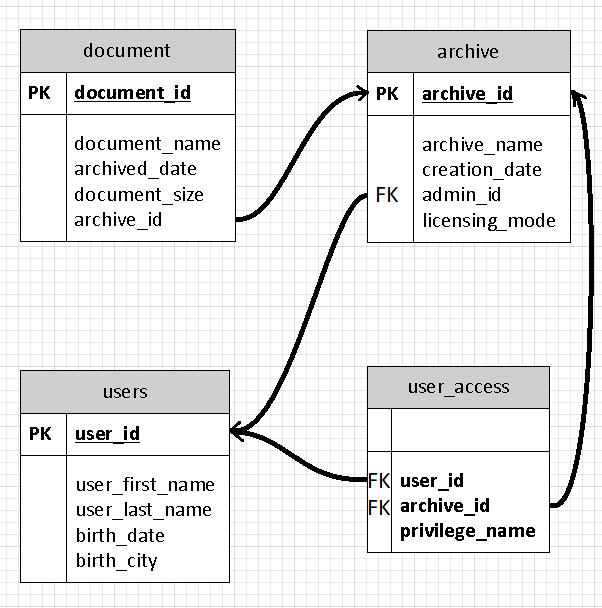
Tabel ”users” – informațiile despre utilizatori și administratori sunt în acest tabel

|  |  |  |  |
| --- | --- | --- | --- |
| Nume Coloana | Tip | Descriere |  |
| user\_id | Char (16) | Id-ul utilizatorului | PK |
| user\_first\_name | Char (32) | Prenumele |  |
| user\_last\_name | Char (32) | Numele de familie |  |
| birth\_city | Char (64) | Localitatea în care sa născut |  |
| birth\_date | datetime | data nașterii |  |

Tabel 4

## Diagrama bazei de date

Diagrama bazei de date:



## Întrebări SQL

1. Câte documente sunt în toate arhivele?
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Selectati numele documentului cu cea mai mare dimensiune (document\_size)
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Selectați id-ul arhivelor care au modul de licențiere (licensing\_mode) per număr document.
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. Cate arhivele sunt în aplicație care conțin mai mult de 1000 de documente
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Selectați id-ul documentelor a căror dimensiune e între 10 și 20 de Megabytes
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. Numele arhivei care contine cele mai multe documente
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
7. Utilizatorii care au acces pe minim 2 arhive și sunt din orașul Chișinău
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
8. Câți useri sunt în fiecare oraș ?
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
9. Selectați numele arhivei si numele administratorului, in cazul in care arhiva nu are un administrator alocat atunci se va afisa valoarea NULL. Indiciu (utilizați LEFT JOIN)
   1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
10. Selectați numele tuturor arhivelor și numele tuturor documentelor întro singură coloana
    1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
11. Selectati numele si prenumele utilizatorilor care au acces la orice arhivă și a căror nume de familie începe cu litera ”A”. Rezultatul nu trebuie să conțină duplicate.
    1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
12. Selectați numele arhivei cu cele mai multe documente
    1. SQL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Întrebări Java

Clasele Java sunt similare tabelelor descrise în capitolele anterioare:

**public** **class** Archive {

**private** String archiveId;

**private** String archiveName;

**private** Date creationDate;

**private** String adminId;

**private** Integer licensingMode;

**private** List<Document> documents;

// Getters and Setters

}

**public** **class** Document {

**private** String documentId;

**private** String documentName;

**private** Date archivedDate;

**private** Integer documentSize;

**private** Archive archive;

// Getters and Setters

}

**public** **class** Access {

**private** User user;

**private** Archive archive;

**private** Integer permission;

// Getters and Setters

}

**public** **class** User {

**private** String userId;

**private** String userFirstName;

**private** String userLastName;

**private** String birthCity;

**private** Date birthDate;

// Getters and Setters

}

### Calcul Model de licențiere

În interiorul clasei Archive completati metodata getLicensingCost, puteti defini si alte metode ajutatoare.

Costul de licențiere va fi calculat în dependență de modul de licențiere a arhivei (licensingMode) :

* Per volumul de date:
  + 1Gb = 5 000 $
  + 100 Gb = 50 000$
  + 1Tb = 100 000$
  + >1Tb, costul fiecarui Tb va fi 0.95% din cel precedent. De exemplu pentru 3 Tb clientul va plăti 100 mii + 95 mii +90,25 mii =285.25 mii dolari
* Per număr de documente:
  + 1000 documente = 200$
  + 50 000 = 5000$
  + 500 000 = 10 mii $
  + 100 000 000 = 50 mii $
  + >100 mln pentru fiecare document ce depaseste 100 000 000 se va achita suma de 0.0005$

**public** Double getLicencingCost(){

}

### Calcul Acces

Completați metoda de mai jos ce va returna valori true sau false daca utilizatorul are sau nu acces la arhiva transmisa ca parametru.

**public** **static** **boolean** isUserAllowed(List<Access> accesses, User user, Archive archive, **int** permission){

}

### Calcul Acces Read

Valorile integer asociate cu privilegiile sunt descrise in tabelul 1.2.3:

Privilegiu:

* 1 - READ (Citire)
* 2 - WRITE (Scriere)
* 3 - DELETE (Ștergere)

Definiți o metodă ce returneaza true sau false daca utilizatorul are dreptul de citire astfel incât în interiorul metodei să fie apelată metoda deja definitia mai sus isUserAllowed

**public** **static** **boolean** isUserAllowedToRead(List<Access> accesses, User user, Archive archive){

**return** **…**;

}

### Numărul arhivelor accesibile de către utilizatori

Definiți metoda ce procesează lista de acces la toate arhivele astfel încât

**public** **static** Map<User,Integer> getAccessorsCount(List<Access> accesses){

**return** **…**;

}

### Definiți clasa CryptoArchive

Aplicația trebuie sa gestioneze Arhive criptate, definiți clasa CryptoArchive ce va avea o proprietate adițională în comparație cu clasa ”Archive” și anume ”cryptoSize” a carui valoare poate fi 128, 256, 512, 1024 de biti – lungimea cheii de criptare.

Lincențierea (metoda *getLicencingCost*) va fi efectuată similar clasei Archive imulțit la (cryptoSize/64).

Definiți noua clasa și noul model de licențiere (metodata *getLicencingCost* suprascrisa):

**public** **class** CryptoArchive

### Adăugați proprietăți și metode adiționale în clasa Archive

Sub-Task 1:

Adăugați o noua proprietate ”archiveContentSize” ce va conține suma dimensiunii tuturor documentelor in arhiva.

Sub-Task 2:

Definiți 2 metode (addDocument, removeDocument) ce adaugă și elimină documente în arhivă, în interiorul metodelor actualizați valoarea ”archivedContentSize”.

Sub-Task 3:

Definiți un nou tip de excepție (o nouă clasă) – DocumentNotFound (completaț liniile cu numele clasei și super clasa de excepție).

**public** **class** \_\_\_\_\_\_\_\_\_\_ **extends** \_\_\_\_\_\_\_\_\_\_ {

}

În interiorul metodei removeDocument excepția va fi aruncată în cazul in care documentul nu a fost identificat în interiorul arhivei. Sub-Task 1,2, 3 și 4 vor fi completate impreuna în listenigul de la final.

Sub-Task 4:

Definiți un nou tip de excepție (o nouă clasă) – DocumentAlreadyExists (completaț liniile cu numele clasei și super clasa de excepție).

**public** **class** \_\_\_\_\_\_\_\_\_\_ **extends** \_\_\_\_\_\_\_\_\_\_ {

}

În interiorul metodei addDocument excepția va fi aruncată în cazul in care documentul există deja în interiorul arhivei.

Sub-Task 1,2, 3 și 4 vor fi completate impreuna în listenigul de la final.

**public** **class** Archive {

**private** String archiveId;

**private** String archiveName;

**private** Date creationDate;

**private** User admin;

**private** Integer licensingMode;

**private** List<Document> documents;

// Getters and Setters Start

…

// Getters and Setters End

// Subtask 1,2,3,4

}

# Java Syntax

## Question 1

Which declare a compilable abstract class? (Choose all that apply.)

A. public abstract class Canine { public Bark speak(); }

B. public abstract class Canine { public Bark speak() { } }

C. public class Canine { public abstract Bark speak(); }

D. public class Canine abstract { public abstract Bark speak(); }

## Question 2

Given:

class Clidders {

public final void flipper() { System.out.println("Clidder"); }

}

public class Clidlets extends Clidders {

public void flipper() {

System.out.println("Flip a Clidlet");

super.flipper();

}

public static void main(String [] args) {

new Clidlets().flipper();

}

}

What is the result?

A. Flip a Clidlet

B. Flip a Clidder

C. Flip a Clidder

Flip a Clidlet

D. Flip a Clidlet

Flip a Clidder

E. Compilation fails.

## Question 3

Given:

class Top {

public Top(String s) { System.out.print("B"); }

}

public class Bottom2 extends Top {

public Bottom2(String s) { System.out.print("D"); }

public static void main(String [] args) {

new Bottom2("C");

System.out.println(" ");

}

}

What is the result?

A. BD

B. DB

C. BDC

D. DBC

E. Compilation fails.

## Question 4

Given the following,

1. class X { void do1() { } }

2. class Y extends X { void do2() { } }

3.

4. class Chrome {

5. public static void main(String [] args) {

6. X x1 = new X();

7. X x2 = new Y();

8. Y y1 = new Y();

9. // insert code here

10. }

11. }

Which, inserted at line 9, will compile? (Choose all that apply.)

A. x2.do2();

B. (Y)x2.do2();

C. ((Y)x2).do2();

D. None of the above statements will compile.

## Question 5

Given:

class Scoop {

static int thrower() throws Exception { return 42; }

public static void main(String [] args) {

try {

int x = thrower();

} catch (Exception e) {

x++;

} finally {

System.out.println("x = " + ++x);

}

}

}

What is the result?

A. x = 42

B. x = 43

C. x = 44

D. Compilation fails.

E. The code runs with no output.

## Question 6

Given:

1. class Crivitch {

2. public static void main(String [] args) {

3. int x = 0;

4. // insert code here

5. do { } while (x++ < y);

6. System.out.println(x);

7. }

8. }

Which, inserted at line 4, produces the output 12?

A. int y = x;

B. int y = 10;

C. int y = 11;

D. int y = 12;

E. int y = 13;

F. None of the above will allow compilation to succeed.

## Question 7

Given:

class Plane {

static String s = "-";

public static void main(String[] args) {

new Plane().s1();

System.out.println(s);

}

void s1() {

try {

s2();

}

catch (Exception e) {

s += "c";

}

}

void s2() throws Exception {

s3(); s += "2";

s3(); s += "2b";

}

void s3() throws Exception {

throw new Exception();

}

}

What is the result?

A. -

B. -c

C. -c2

D. -2c

E. -c22b

F. -2c2b

G. -2c2bc

H. Compilation fails.

## Question 8

Consider the following program:

interface Side { String getSide(); }

class Head implements Side {

public String getSide() { return "Head "; }

}

class Tail implements Side {

public String getSide() { return "Tail "; }

}

class Coin {

public static void overload(Head side) { System.out.print(side.getSide()); }

public static void overload(Tail side) { System.out.print(side.getSide()); }

public static void overload(Side side) { System.out.print("Side "); }

public static void overload(Object side) { System.out.print("Object "); }

public static void main(String []args) {

Side firstAttempt = new Head();

Tail secondAttempt = new Tail();

overload(firstAttempt);

overload((Object)firstAttempt);

overload(secondAttempt);

overload((Side)secondAttempt);

}

}

What is the output of this program when executed?

a) Head Head Tail Tail

b) Side Object Tail Side

c) Head Object Tail Side

d) Side Head Tail Side

## Question 9

Consider the following program:

class Base {

public static void foo(Base bObj) {

System.out.println("In Base.foo()");

bObj.bar();

}

public void bar() {

System.out.println("In Base.bar()");

}

}

class Derived extends Base {

public static void foo(Base bObj) {

System.out.println("In Derived.foo()");

bObj.bar();

}

public void bar() {

System.out.println("In Derived.bar()");

}

}

class OverrideTest {

public static void main(String []args) {

Base bObj = new Derived();

bObj.foo(bObj);

}

}

Which is the putput of the program:

a)

In Base.foo()

In Base.bar()

b)

In Base.foo()

In Derived.bar()

c)

In Derived.foo()

In Base.bar()

d)

In Derived.foo()

In Derived.bar()

## Question 10

Consider the following program:

class EHBehavior {

public static void main(String []args) {

try {

int i = 10/0; // LINE A

System.out.print("after throw -> ");

} catch(ArithmeticException ae) {

System.out.print("in catch -> ");

return;

} finally {

System.out.print("in finally -> ");

}

System.out.print("after everything");

}

}

Which one of the following options best describes the behavior of this program?

* 1. The program prints the following: in catch -> in finally -> after everything.
  2. The program prints the following: after throw -> in catch -> in finally -> after everything.
  3. The program prints the following: in catch -> in finally -> after everything.
  4. The program prints the following: in catch -> after everything.
  5. The program prints the following: in catch -> in finally ->.
  6. When compiled, the program results in a compiler error in line marked with comment in LINE A for divide-by-zero.

## Question 11

Given:

TreeSet map = new TreeSet();

map.add("one"); map.add("two"); map.add("three"); map.add("four"); map.add("one");

Iterator it = map.iterator();

while (it.hasNext() ) {

System.out.print( it.next() + " " );

}

What is the result?

A. Compilation fails.

B. one two three four

C. four three two one

D. four one three two

E. one two three four one

F. one four three two one

G. An exception is thrown at runtime.

H. The print order is not guaranteed.

## Question 12

Given the following:

interface I1 {}

interface I2 {}

abstract class C {}

which one of the following will compile without errors?

1. class CI12 extends C, I1, I2 {}
2. class CI12 implements C extends I1, I2 {}
3. class CI12 implements C, I1, I2 {}
4. class CI12 extends C implements I1, I2 {}
5. class CI12 extends C implements I1 implements I2 {}
6. class CI12 implements C extends I1 extends I2 {}