

Given the following declarations in Scala:

```
class A { def foo(i: Int) = print(i) }
```

```
trait B extends A { override def foo(i: Int) = super.foo( i * 2 ) }
```

```
trait C extends A { override def foo(i: Int) = super.foo( i + 1 ) }
```

```
trait D extends A { override def foo(i: Int) = super.foo( i * i ) }
```

```
val x = new A with B with C with D
```

What is the value printed by the call **x.foo(7)** ?

Câu trả lời:

100

Given the following code in Python:

```
class M:
    def foo(self,i):
        print(i * 2)
class N(M):
    pass

class Q(N):
    def foo(self,i):
        print(i * i)
```

What is the printed result of the following code?

```
x = Q()
```

```
N.foo(x,4)
```

Câu trả lời:

8

Given the following declarations in a static type checking object-oriented programming language:

```
class A { def foo() = print("a") }
```

```
class B extends A { } // B is a subclass of A
```

```
class C extends B { override def foo() = print("c") } // C is a subclass of B
```

```
class D extends A { override def foo() = print("d") } // D is a subclass of A
```

Assume that variable `b` is declared in type `B` and is referred to some object, what value can be printed by the call **`b.foo()`** in the corresponding object referred by variable `b`?

Chọn một hoặc nhiều hơn:

- ☒ a. a (if `b` is referred to a `B` object)
- ☐ b. c (if `b` is referred to a `C` object)
- ☒ c. a (if `b` is referred to an `A` object)
- ☐ d. d (if `b` is referred to a `D` object)

Given that class `A` is the super class of class `B` and the following declarations and initializations in a general static-type checking object-oriented programming language:

```
A x = new B(); //a
```

```
B y = new A(); //b
```

Select the correct choice?

Chọn một:

- ☐ a. both are correct
- ☐ b. both are wrong
- ☐ c. Statement `//a` is wrong but statement `//b` is correct
- ☒ d. Statement `//a` is correct but statement `//b` is wrong

Given the following Scala fragment code,

```
class SpecialNum(val x:Int)
```

```
trait Add extends SpecialNum { def +(other:SpecialNum) = x + other.x }
```

```
trait Mul extends SpecialNum { def *(other:SpecialNum) = x * other.x }
```

```
val x = new SpecialNum(3) with Add
```

```
val y = new SpecialNum(5) with Mul
```

Fill in the blanks such that expressions $x + y$ and $y * x$ are valid while $y + x$ and $x * y$ are invalid. Make sure that there is only one space in each blank.

Please select the APPROPRIATE static field when defining class CLASS in a school?

Chọn một hoặc nhiều hơn:

- ☐ a. int number; // the number of student in the class
- ☐ b. String name; // the name of the lecture in charge of the class
- ☒ c. int maxNo; // the maximum number of students are allowed in a class
- ☒ d. int classCount; // the number of classes in the school

Assume class A is declared in package **example** using Scala. Write the **access modifier** in the blank such that the corresponding member can be accessed only in package example? Write word **empty** if you think it does not need to write anything.

Trả lời:

Given the following Python code:

```
class A:
```

```
    def foo(self,i): print (i)
```

```
class B(A):
```

```
    def foo(self,i): super().foo(i * 2)
```

```
class C(A):
```

```
    def foo(self,i): super().foo(i + 1)
```

```
class D(A):
```

```
    def foo(self,i): super().foo(i * i)
```

```
class E(  ): pass
```

```
x = E()
```

```
x.foo(3) // printed value is 37 i.e. ((3*2)*(3*2))+1
```

Fill the superclass of E in the blank such that the expression x.foo(3) gives the expected result (37)

B, D, C

“B, D, C”

Given the following declarations in Scala:

```
class A { def foo(i: Int) = print(i) }
```

```
trait B extends A { override def foo(i: Int) = super.foo( i + 1 ) }
```

```
trait C extends A { override def foo(i: Int) = super.foo( i * 2 ) }
```

```
trait D extends A { override def foo(i: Int) = super.foo( i * i ) }
```

To make x.foo(10) calculate as (((10 * 2) * (10 * 2)) + 1), please match (1), (2) and (3) in the following declaration?

```
val x = new A with (1) with (2) with (3)
```

C → D ⇒ B

B D C

(2)	<input type="text" value="C"/>	↕
(1)	<input type="text" value="D"/>	↕
(3)	<input type="text" value="B"/>	↕

D
B
C

(2) D

(1) C

(3) B