CSPB 3155 - Reckwerdt - Principles of Programming Languages

Dashboard / My courses / 2244:CSPB 3155 / Week 11: Lazy Evaluation, Streams, and Iterators / Spot Exam 5

Started on	Thursday, 1 August 2024, 2:47 PM
State	Finished
Completed on	Thursday, 1 August 2024, 3:00 PM
Time taken	13 mins 25 secs
Grade	**8.50** out of 10.00 (**85**%)

Question **1**

Correct

Mark 2.00 out of 2.00

[4 points] OBJECTS

(A: 2 Points) Consider the following code:

```
class Alpha
class Bravo extends Alpha
class Charlie

class Delta extends Bravo with Charlie
```

The above code does not work in Scala. Suppose the developer is attempting to use Charlie as a Mix-In for class Delta, then what is wrong with the above code? (select one)

○ Nothing is wrong with the code

○ Classes Alpha, Bravo and Charlie must all be abstract

● Charlie must be a trait ✓

○ Charlie must be an abstract class

○ the class Alpha must be abstract

Mark 2.00 out of 2.00

The correct answer is: Charlie must be a trait

(B: 2 points) Consider the following code

```
class Alpha { def foo = { println("xkcd") } }
class Bravo extends Alpha { override def foo = { println("wat?") } }

def foobalizer(a:Alpha) = { a.foo() }

foobalizer(new Bravo)
```

What is printed in the above code? (select one)

○ xkcd

○ This code has a bug and it will not run

○ This code will not print anything because println is never called

● wat? ✓

Mark 2.00 out of 2.00

The correct answer is: wat?

Question **2**

Correct

Mark 2.00 out of 2.00

[9 points] EXPLICIT TYPE CHECKING

(A: 2 Points) Consider the following Lettuce Program

```
let f : num => (num => num) =
                     function (x :num)
                          function (y : num)
                                   x + y
in
    f(2)
```

What is the type of the (value returned by) above program? (select one)

○ num => bool

● num => num ✓

○ num

○ bool

○ Type Error

Mark 2.00 out of 2.00

The correct answer is: num => num

(B: 2 Points) Consider the following Lettuce Program

```
let f : num => (num => num) =
                     function (x :num)
                          function (y : num)
                                   x + y
in
      f(2)(5)
```

What is the type of the above program? (select one)

○ num => bool

● num ✓

○ bool

○ Type Error

○ num => num

Mark 2.00 out of 2.00

The correct answer is: num

(C: 2 Points) Consider the following Lettuce Program

```
  let f : (num => bool) => num => num =
     function (g :num => bool)
          function (y : num)
              g(y)
   in
      f(2)
```

What is the type of this program? (select one)

● Type Error ✓

○ num => bool

○ num => num

○ num

○ bool

Mark 2.00 out of 2.00

The correct answer is: Type Error

(D: 3 Points) Consider the following Lettuce Program

```
let f : (num => bool) => num => num =
    function (g :num => bool)
        function (y : num)
            g(y)
in
    f (function (x: num) x > 2) (3)
```

What is the type of this program? (select one)

○ num

○ num => bool

○ bool

● Type Error ✓

○ num => num

Mark 3.00 out of 3.00

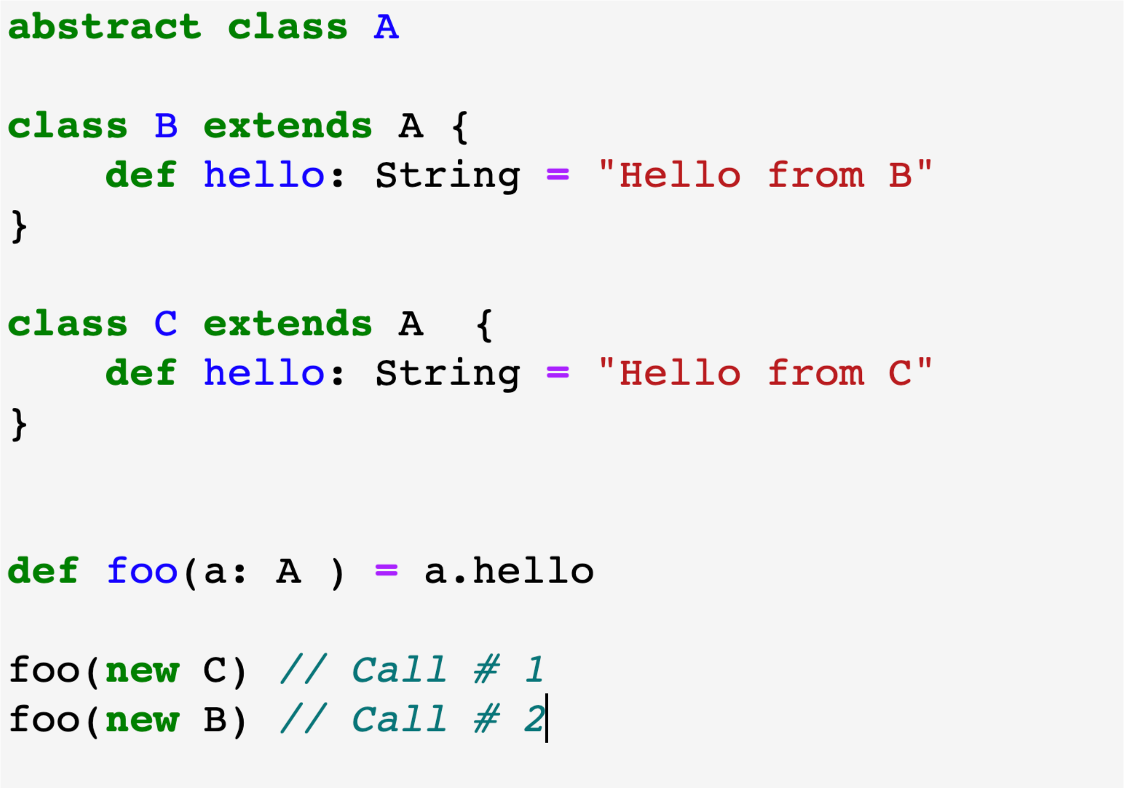
The correct answer is: Type Error

Question **3**

Partially correct

Mark 1.50 out of 3.00

Consider the code snippet below:



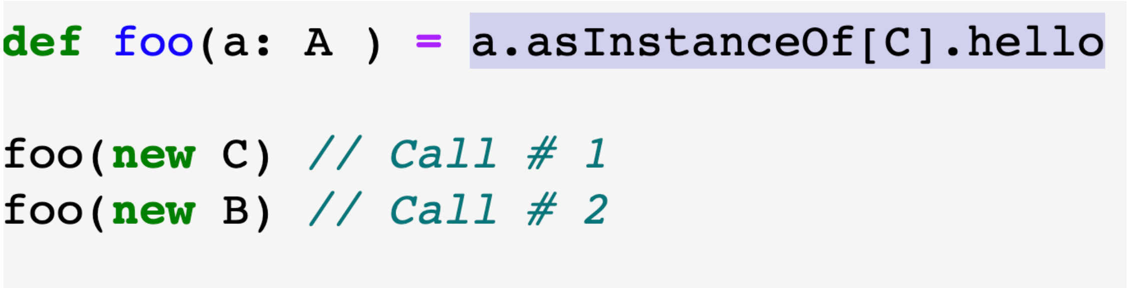
```
abstract class A

class B extends A {
    def hello: String = "Hello from B"
}

class C extends A  {
    def hello: String = "Hello from C"
}


def foo(a: A ) = a.hello

foo(new C) // Call # 1
foo(new B) // Call # 2
```

Select the true statement from the choices below:

□ This code fails to compile.

☑ This code compiles just fine since the calls to foo only involve classes that have the "hello" method implemented ✖

Suppose we modified the definition of foo as follows while keeping classes `A`, `B`, and `C` unchanged.



```
def foo(a: A ) = a.asInstanceOf[C].hello

foo(new C) // Call # 1
foo(new B) // Call # 2
```

Recall "asInstanceOf[...]" attempts to downcast a to an object of type ".....".

Select the true statement from the dropdown menu:

☑ The code compiles, Call #1 returns without an error whereas Call # 2 throws an exception. ✔

□ The code fails to compile

□

The code compiles, however Call # 1 throws an exception.

Your answer is partially correct.

1 of your answers is correct.

Question **4**

Correct

Mark 3.00 out of 3.00

Consider the following scala code snippet

```
trait Animal {
    def sound(emotion: String): String
    val genus: String
    val species: String
}

class Zoo[T] ( allAnimals: List[T]) {
    def allSounds(emotion: String) = allAnimals.map(_.sound(emotion))
    def allGenera: List[String] = allAnimals.map(_.genus)
    def allSpecies: List[String] = allAnimals.map(_.species)
}
```

(A) Select what happens when you try to run this coded snippet:

□

It compiles but throws an exception when we attempt to run it

☑

It causes a compilation error ✓

□

It compiles and runs without any errors.

(B) What is wrong (if anything) with the code above?

□

T must be replaced everywhere by Animal.

□

Nothing: it compiles and runs without errors.

☑

There is no guarantee that the type T has methods sound and fields/methods genus and species ✓

(C) Which of the suggested fixes below will make the code compile and run?

□

Replace T by Animal everywhere.

□

Trick question: it compiles and runs just fine.

☑

Add the type constraint T <: Animal ✓

Your answer is correct.