Prep 3 quiz: Inheritance

Started: Sep 24 at 9:50pm

Quiz Instructions

Note: this is only one part of this week's prep! Please find the prep handout at
https://www.teach.cs.toronto.edu/~csc148h/fall/preps/prep3/handout/prep3.html
(https://www.teach.cs.toronto.edu/~csc148h/fall/preps/prep3/handout/prep3.html) .

Question 1 — 1 pts

A library contains books and DVDs, which can be checked out and returned. A warning is sent to borrowers who have overdue items. Books can be renewed but DVDs cannot, and DVDs have a shorter loan period than books. The library may expand in future to include other kinds of items.

Suppose you are designing classes for the library. Which of the designs below is most appropriate?

- ○ A Book class and a DVD class that are unrelated, and a LibraryItem class that stores an instance of Book or DVD in an instance attribute
- ○ A LibraryItem class whose attributes can be initialized differently depending on whether it represents a book or DVD
- ○ A Book class and a DVD class that have no relationship
- ◉ A LibraryItem parent class that stores common information such as due date, and child classes Book and DVD.

Question 2 — 1 pts

Suppose we have a Parent class that has a method that is not completed because child classes have different implementations. Fill in the blank in the code below.

```
class Parent:
    # docstring, __init__ omitted
    def method(self) -> None:
```

```
        """A method that differs for each of the child classes.
        """
        raise _______________
```

NotImplementedError

Question 3 1 pts

Which of the following is true about abstract classes?

- ○ Abstract classes should only be instantiated when a more specific subclass does not exist
- ◉ Abstract classes should never be instantiated
- ○ It is acceptable to instantiate an abstract class in any situation
- ○ An abstract class can be instantiated if it is likely that the abstract methods will never be called

Question 4 1 pts

Given the code below, which of the following statements could go in the main block and would execute without raising an error?

```
class Mystery:
    """A mystery class without any implementation.
    """
    pass

if __name__ == '__main__':
```

- ☐ None of the above. They would all raise an error.

```
m = Mystery('what is this?')
```

```
m = Mystery()
print(m)
```

- [x]
```
m = Mystery()
```

Question 5 1 pts

What would be the result of running the code below?

```
class Parent:
    """An arbitrary class
    """
    num1: int

    def __init__(self, num1: int) -> None:
        self.num1 = num1


class Child(Parent):
    """A subclass.
    """
    num2: int

    def __init__(self, num1: int, num2: int) -> None:
        self.num2 = num2


if __name__ == '__main__':
    c = Child(1, 2)
    print(c.num2)
    print(c.num1)
```

```
2
Traceback (most recent call last):
...
AttributeError ...
```

```
2
0
```

```
Traceback (most recent call last):
...
AttributeError ...
```

```
2
1
```

Question 6 1 pts

An animal has attributes such as species and age. A mammal is a kind of animal that has additional attributes such as the colour of its fur (all mammals have fur/hair).

We have started some code for an `Animal` class and a `Mammal` subclass, and now we want to implement initializers. Which code is the most appropriate?

```
class Animal:
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int) -> None:
        """Initialize this Animal.

        Precondition: age >= 0
        """
        self.species = species
        self.age = age

class Mammal(Animal):
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int, fur_colour: str) -> None:
        """Initialize this Mammal.
        """
        Animal.__init__(self, species, age)
        self.species = species
        self.age = age
        self.fur_colour = fur_colour
```

```
class Animal:
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int) -> None:
        """Initialize this Animal.

        Precondition: age >= 0
        """
        self.species = species
        self.age = age

class Mammal(Animal):
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int, fur_colour: str) -> None:
        """Initialize this Mammal.
        """
        Animal.__init__(self, species, age)
        self.fur_colour = fur_colour
```

```
class Animal:
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int) -> None:
        """Initialize this Animal.

        Precondition: age >= 0
        """
        self.species = species
        self.age = age

class Mammal(Animal):
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int, fur_colour: str) -> None:
        """Initialize this Mammal.
        """
        self.species = species
        self.age = age
        self.fur_colour = fur_colour
```

```
class Animal:
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int) -> None:
        """Initialize this Animal.

        Precondition: age >= 0
        """
        self.species = species
        self.age = age

class Mammal(Animal):
    """DOCSTRING OMITTED"""
    def __init__(self, species: str, age: int, fur_colour: str) -> None:
        """Initialize this Mammal.
        """
        Animal.__init__(self)
        self.fur_colour = fur_colour
```

Question 7 — 1 pts

What will be the output of the three calls to `print_message` at the end of this program?

```
class Printer:
```

```
    """A Printer.

    === Attributes ===
    message: the message to print
    """
    message: str

    def __init__(self, message: str) -> None:
        self.message = message

    def print_message(self):
        """Prints a message.
        """
        print(self.message)


class MysteryPrinter(Printer):
    def __init__(self, message: str) -> None:
        Printer.__init__(self, message + message)


class DoublePrinter(MysteryPrinter):
    def print_message(self):
        print(self.message + self.message)


if __name__ == '__main__':
    m1 = Printer('hello')
    m2 = MysteryPrinter('good')
    m3 = DoublePrinter('bye')
    m1.print_message()
    m2.print_message()
    m3.print_message()
```

Line 1:

hello

Line 2:

goodgood

Line 3:

byebyebyebye

Question 8 1 pts

Given the code below, which of the following statements will evaluate to True?

```
class F:
    # Implementation Omitted

class G(F):
    # Implementation Omitted

class H(F):
    # Implementation Omitted

if __name__ == '__main__':
    f = F()
    g = G()
    h = H()
```

- ☐ isinstance(g, H)
- ☑ isinstance(h, F)
- ☐ type(f) == type(g)
- ☐ type(g) == type(h)
- ☑ isinstance(g, F)
- ☐ isinstance(h, G)

Question 9 1 pts

Given the code below, which of the following statements will evaluate to True?

```
class Machine:
    # implementation omitted

class Calculator(Machine):
    # implementation omitted

class FourFunctionCalculator(Calculator):
    # implementation omitted
```

```
if __name__ == '__main__':
    m = Machine()
    c = Calculator()
    f = FourFunctionCalculator()
```

- ☐ isinstance(m, Calculator)
- ☑ isinstance(c, Machine)
- ☐ type(m) == type(c)
- ☐ type(f) == type(c)
- ☑ isinstance(f, Calculator)
- ☑ isinstance(f, Machine)
- ☐ type(m) == type(f)

Quiz saved at 10:10pm Submit Quiz