1. Abstract - Quiz

Q1 of 6

What is the output of the following code snippet?

```
from abc import ABCMeta, abstractmethod
class Parent(metaclass=ABCMeta):
    def __init__(self):
        self.num=100

    @abstractmethod
    def show(self):
        pass

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.__var=10

    def show(self):
        print(self.num)
        print(self.__var)

obj=Parent()
obj.show()
```

a) 100
10

b) 10
100

c) Error: abstract method should always have a valid statement other than pass

d) Error: abstract class cannot be instantiated

- ○ a
- ○ b
- ○ c
- ◉ d ✓

Q2 of 6

What is the output of the following code snippet?

```
from abc import ABCMeta, abstractmethod
class Parent(metaclass=ABCMeta):
    def __init__(self):
        self.num=5
    @abstractmethod
    def show(self):
        pass

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.__var=10
    def show(self):
        print(self.num)
        print(self.__var)


obj=Child()
obj.show()
```

a) 10
5

b) 5
10

c) Error: an abstract method cannot be overridden in child class

d) Error: class Child should be abstract.

- ○ a
- ◉ b ✓
- ○ c
- ○ d

Q3 of 6

What is the output of the following code snippet?

```
from abc import ABCMeta, abstractmethod
class Parent(metaclass=ABCMeta):
    def __init__(self):
        print(100)

    @abstractmethod
    def show(self):
        pass

class Child(Parent):
    def __init__(self):
        super().__init__()
        print(10)

obj=Child()
obj.show()
```

a) 100
10

b) Error: abstract class cannot be instantiated

- ○ a
- ◉ b ✓

Q4 of 6

What is the output of the following code snippet?

```
from abc import ABCMeta, abstractmethod
class Parent(metaclass=ABCMeta):
    def __init__(self):
        self.num=5
    @abstractmethod
    def show(self):
        pass

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.var=10

class GrandChild(Child):
    def show(self):
        print(self.num)
        print(self.var)
        print("This is possible")

obj=GrandChild()
obj.show()
```

a) 5
10
This is possible

b) 10
5
This is possible

c) Error: Child class should override abstract method show() of Parent class

d) Error: Child class should be declared as abstract

- (●) a ✓
- ( ) b
- ( ) c
- ( ) d

Q5 of 6

What will be the output of the code given below?

```
from abc import ABCMeta, abstractmethod
class A(metaclass=ABCMeta):
    def __init__(self):
        print("123")
    @abstractmethod
    def method1(self):
        pass
class B(A):
    def method2(self):
        print("456")
obj=B()
obj.method2()
```

a) 123
456

b) 123

c) 456

d) Error: class B cannot be instantiated

- ○ a
- ○ b
- ○ c
- ◉ d ✓

Q6 of 6

What kind of relationship is listed below?

```
class Trade:
    def __init__(self):
        self.__trade_detail = None

    def get_trade_detail(self):
        return self.__trade_detail

    def set_trade_detail(self, value):
        self.__trade_detail = value

class TradeDetail:
    def __init__(self):
        self.__trade_id = None
        self.__order_id = None

    def get_trade_id(self):
        return self.__trade_id

    def get_order_id(self):
        return self.__order_id

    def set_trade_id(self, value):
        self.__trade_id = value

    def set_order_id(self, value):
        self.__order_id = value

t=Trade()
trade_detail=TradeDetail()
trade_detail.set_trade_id(10)
trade_detail.set_order_id(42)
t.set_trade_detail(trade_detail)
```

- (●) Aggregation ✓
- ( ) Inheritance
- ( ) Abstract class
- ( ) None of the above

2. Exercise on Abstract - Level 2

Answer Link -

https://github.com/bsccs2023/Naan-Mudhalvan-2024/blob/90979c8ba53d8a4ae2478e5e4789801cb22fe932/10--Abstract/2--Exercise-on-Abstract-Level-2.py