Abbottabad University Of Science And Technology

Department Of Computer Science

ASSIGNMENT NO 5

Name : **Shayan Turk**

Roll No : **2023139**

Semester : **2nd**

Subject : **OOP**

Submitted To : **Sir jamal Abdul Ahad**

Question No 1

```
class LoggingMeta(type):     def
__new__(cls, name, bases, attrs):
print(f"Creating class {name}")
       return super().__new__(cls, name, bases, attrs)

    def __init__(cls, name, bases, attrs):
print(f"Initializing class {name}")
       super().__init__(name, bases, attrs)

# Example usage:
class MyClass(metaclass=LoggingMeta):
def __init__(self):
       print("Instance created.")

# Output:
# Creating class MyClass
# Initializing class MyClass
```

Question No 2

```
class SingletonMeta(type):
   _instances = {}

   def __call__(cls, *args, **kwargs):        if cls not
in cls._instances:            instance =
super().__call__(*args, **kwargs)
          cls._instances[cls] = instance
       return cls._instances[cls]

# Example usage: class
SingletonClass(metaclass=SingletonMeta):
   pass

obj1 = SingletonClass() obj2
= SingletonClass()

print(obj1 is obj2)  # Output: True
```

Question No 3

```
class AttributeValidationMeta(type):     def
__new__(cls, name, bases, attrs):        for
attr_name, attr_value in attrs.items():
if isinstance(attr_value, int):              if
attr_value < 0:
                raise ValueError(f"Attribute {attr_name} must be a non-negative integer.")
return super().__new__(cls, name, bases, attrs)

# Example usage: class
ValidatedClass(metaclass=AttributeValidationMeta):
   positive_integer_attr = 10
   negative_integer_attr = -5  # Raises ValueError

# Output:

# ValueError: Attribute negative_integer_attr must be a non-negative integer.
```

Question No 4

```
class MultipleInheritanceMeta(type):
def __new__(cls, name, bases, attrs):
      # Custom logic for managing multiple inheritance
if len(bases) > 1:
         raise ValueError(f"Multiple inheritance not allowed for class {name}")
return super().__new__(cls, name, bases, attrs)

# Example usage: class
Base1:
   pass

class Base2:
   pass

class MyClass(Base1, Base2, metaclass=MultipleInheritanceMeta):
   pass  # Raises ValueError

# Output:
# ValueError: Multiple inheritance not allowed for class MyClass
```