Assignment

Abbottabad University Of Science And Technology

Departement Of Computer Science

Name : M Hassan Ashraf

Roll No : 2023132

Subject : OOP

Class : CS – 2nd A

Submitted to:

Sir Jamal Abdul Ahad

Q 1: Attribute Validation Metaclass: Design a metaclass that performs custom validation on attributes declared during class creation.

```
class AttributeValidationMeta(type):
    def __new__(cls, name, bases, attrs):

        for attr_name, attr_value in attrs.items():
            if isinstance(attr_value, str) and len(attr_value) < 5:
                raise ValueError(f"Attribute '{attr_name}' must have a minimum length of 5 characters.")


        return super().__new__(cls, name, bases, attrs)

class MyClass(metaclass=AttributeValidationMeta):
    attribute1 = "valid_value"
    attribute2 = "short”
```

Q2: Hooking Class Creation: Build a metaclass that intercepts specific steps of class creation (e.g., attribute addition, method creation) and injects custom logic.

```
class CustomLogicMeta(type):
    def __new__(cls, name, bases, attrs):

        for attr_name, attr_value in attrs.items():
            if callable(attr_value):

                original_method = attrs[attr_name]
                def wrapper(*args, **kwargs):

                    print(f"Custom logic before calling {attr_name}")
                    result = original_method(*args, **kwargs)
                    print(f"Custom logic after calling {attr_name}")
                    return result
                attrs[attr_name] = wrapper

        return super().__new__(cls, name, bases, attrs)
```

```
class MyClass(metaclass=CustomLogicMeta):
    def method1(self):
        print("Executing method1")

    def method2(self):
        print("Executing method2")

obj = MyClass()
obj.method1()
obj.method2()
```

Q3. Mixin Classes: Create Mixin classes that provide specific functionalities (e.g., logging, serialization) and utilize them through multiple inheritance.

```
class LoggingMixin:
    def log(self, message):
        print(f"Log: {message}")


class SerializationMixin:
    def serialize(self):
        return str(self.__dict__)


class MyClass(LoggingMixin, SerializationMixin):
    def __init__(self, name, value):
        self.name = name
        self.value = value

obj = MyClass(name="example", value=42)

obj.log("This is a log message")

serialized_data = obj.serialize()
print(f"Serialized Data: {serialized_data}")
```