File content analysis

Diamond inheritance problem

• BaseClass is a common base class for [**CorrectSubclass**, **SubclassWithoutSuper**]

• Employee is a common base class for [**Developer**, **Manager**]

Not using the super() function

• The Employee class inherits from [**ABC**] and does not use super() in the constructor.

• The Developer class inherits from [**Employee**] and does not use super() in the constructor.

• The Manager class inherits from [**Employee**] and does not use super() in the constructor.

• The Mammal class inherits from [**Animal**] and does not use super() in the constructor.

• The Primate class inherits from [**Mammal**] and does not use super() in the constructor.

• The Human class inherits from [**Primate**] and does not use super() in the constructor.

• The Person class inherits from [**Human**] and does not use super() in the constructor.

• The SubclassWithoutSuper class inherits from [**BaseClass**] and does not use super() in the constructor.

Solutions

• To resolve diamond inheritance involving **BaseClass**, ensure all derived classes use cooperative multiple inheritance with **super()**. This approach ensures each class in the hierarchy is initialized exactly once. Below is an example of how to correctly implement constructors in derived classes: class CorrectSubclass(BaseClass): def __init__(self, *args, **kwargs): super().__init__(*args, **kwargs) # Correctly initialize parent classes # Initialize any additional attributes for this class here class SubclassWithoutSuper(BaseClass): def __init__(self, *args, **kwargs): super().__init__(*args, **kwargs) # Correctly initialize parent classes # Initialize any additional attributes for this class here Additionally, consider redesigning the class hierarchy to reduce complexity or ambiguity, potentially eliminating the need for multiple inheritance.

• To resolve diamond inheritance involving **Employee**, ensure all derived classes use cooperative multiple inheritance with **super()**. This approach ensures each class in the hierarchy is initialized exactly once. Below is an example of how to correctly implement constructors in derived classes: class Developer(Employee): def __init__(self, *args, **kwargs): super().__init__(*args, **kwargs) # Correctly initialize parent classes # Initialize any additional attributes for this class here class Manager(Employee): def __init__(self, *args, **kwargs): super().__init__(*args, **kwargs) # Correctly initialize parent classes # Initialize any additional attributes for this class here Additionally, consider redesigning the class hierarchy to reduce complexity or ambiguity, potentially eliminating the need for multiple inheritance.

Static variable issue

• Class **DatabaseHandler** defines a static variable **query**, which may cause side effects if modified across instances.

• Class **DatabaseHandler** defines a static variable **cursor**, which may cause side effects if modified across instances.

• Class **DatabaseHandler** defines a static variable **result**, which may cause side effects if modified across instances.

• Class **HermetyzacjaVisitor** defines a static variable **tab_helper**, which may cause side effects if modified across instances.

• Class **HermetyzacjaVisitor** defines a static variable **has_public**, which may cause side effects if modified across instances.

• Class **HermetyzacjaVisitor** defines a static variable **has_private**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **max_depth**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **var_name**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **base_methods**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **inheritance_depth**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **bases**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **derived_methods**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **new_depth**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **tree**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **content**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **common_bases**, which may cause side effects if modified across instances.

• Class **DependencyMapper** defines a static variable **base_classes**, which may cause side effects if modified across instances.

• Class **User** defines a static variable **shared_data**, which may cause side effects if modified across instances.

Complex inheritance hierarchy

• Class **Primate** has a complex inheritance hierarchy with depth of 3, which may complicate the codebase.

• Class **Human** has a complex inheritance hierarchy with depth of 4, which may complicate the codebase.

• Class **Person** has a complex inheritance hierarchy with depth of 5, which may complicate the codebase.

Missing abstract method implementations

• Class **Developer** does not implement all abstract methods from its base class **Employee**. Missing methods:

report

• Class **Manager** does not implement all abstract methods from its base class **Employee**. Missing methods:

work

Potential SQL Injection vulnerabilities found in file: ../../additional\helping_code.py,

Line 182: BinOp(left=Name(id=**current_depth**, ctx=Load()), op=Add(), right=Constant(value=1, kind=None))

Potential SQL Injection vulnerabilities found in file: ../../additional\insecure_deserialization.py,

Line 182: BinOp(left=Name(id=**current_depth**, ctx=Load()), op=Add(), right=Constant(value=1, kind=None))

Potential SQL Injection vulnerabilities found in file: ../../additional\subclass_without_super.py,

Line 182: BinOp(left=Name(id=**current_depth**, ctx=Load()), op=Add(), right=Constant(value=1, kind=None))

Potential SQL Injection vulnerabilities found in file: ../../additional\test_csrf.py,

Line 182: BinOp(left=Name(id=**current_depth**, ctx=Load()), op=Add(), right=Constant(value=1, kind=None))

Potential SQL Injection vulnerabilities found in file: ../../additional\user.py,

Line 182: BinOp(left=Name(id=**current_depth**, ctx=Load()), op=Add(), right=Constant(value=1, kind=None))

Potential SQL Injection vulnerabilities found in file: ../../additional\xss_detector_vulnerabilities.py,

Line 182: BinOp(left=Name(id=**current_depth**, ctx=Load()), op=Add(), right=Constant(value=1, kind=None))

Potential CSRF vulnerabilities found in file: ../../additional\test_csrf.py,

Line 7: FunctionDef(name=**submit**, args=arguments(posonlyargs=[], args=[], vararg=None, kwonlyargs=[],

kw_defaults=[], kwarg=None, defaults=[]), body=[Assign(targets=[Name(id=**data**, ctx=Store())],

value=Subscript(value=Attribute(value=Name(id=**request**, ctx=Load()), attr=**form**, ctx=Load()),

slice=Index(value=Constant(value=**data**, kind=None)), ctx=Load()), type_comment=None),

Expr(value=Call(func=Name(id=**print**, ctx=Load()), args=[Constant(value=**Data received:**, kind=None),

Name(id=**data**, ctx=Load())], keywords=[])), Return(value=Constant(value=**Data processed**, kind=None))],

decorator_list=[Call(func=Attribute(value=Name(id=**app**, ctx=Load()), attr=**route**, ctx=Load()),

args=[Constant(value=**/submit**, kind=None)], keywords=[keyword(arg=**methods**,

value=List(elts=[Constant(value=**POST**, kind=None)], ctx=Load()))])], returns=None, type_comment=None)

Potential CSRF vulnerabilities found in file: ../../additional\user.py,

Line 7: FunctionDef(name=**submit**, args=arguments(posonlyargs=[], args=[], vararg=None, kwonlyargs=[],

kw_defaults=[], kwarg=None, defaults=[]), body=[Assign(targets=[Name(id=**data**, ctx=Store())],

value=Subscript(value=Attribute(value=Name(id=**request**, ctx=Load()), attr=**form**, ctx=Load()),

slice=Index(value=Constant(value=**data**, kind=None)), ctx=Load()), type_comment=None),

Expr(value=Call(func=Name(id=**print**, ctx=Load()), args=[Constant(value=**Data received:**, kind=None),

Name(id=**data**, ctx=Load())], keywords=[])), Return(value=Constant(value=**Data processed**, kind=None))],

decorator_list=[Call(func=Attribute(value=Name(id=**app**, ctx=Load()), attr=**route**, ctx=Load()),

args=[Constant(value=**/submit**, kind=None)], keywords=[keyword(arg=**methods**,

value=List(elts=[Constant(value=**POST**, kind=None)], ctx=Load()))])], returns=None, type_comment=None)

Potential CSRF vulnerabilities found in file: ../../additional\xss_detector_vulnerabilities.py,

Line 7: FunctionDef(name=**submit**, args=arguments(posonlyargs=[], args=[], vararg=None, kwonlyargs=[],

kw_defaults=[], kwarg=None, defaults=[]), body=[Assign(targets=[Name(id=**data**, ctx=Store())],

value=Subscript(value=Attribute(value=Name(id=**request**, ctx=Load()), attr=**form**, ctx=Load()),

slice=Index(value=Constant(value=**data**, kind=None)), ctx=Load()), type_comment=None),

Expr(value=Call(func=Name(id=**print**, ctx=Load()), args=[Constant(value=**Data received:**, kind=None),

Name(id=**data**, ctx=Load())], keywords=[])), Return(value=Constant(value=**Data processed**, kind=None))],

decorator_list=[Call(func=Attribute(value=Name(id=**app**, ctx=Load()), attr=**route**, ctx=Load()),

args=[Constant(value=**/submit**, kind=None)], keywords=[keyword(arg=**methods**,

value=List(elts=[Constant(value=**POST**, kind=None)], ctx=Load()))])], returns=None, type_comment=None)

Potential Insecure Deserialization vulnerabilities found in file: ../../additional\insecure_deserialization.py,

Line 7: Call(func=Attribute(value=Name(id=**pickle**, ctx=Load()), attr=**loads**, ctx=Load()), args=[Name(id=**data**,

ctx=Load())], keywords=[])

Potential Insecure Deserialization vulnerabilities found in file: ../../additional\subclass_without_super.py,

Line 7: Call(func=Attribute(value=Name(id=**pickle**, ctx=Load()), attr=**loads**, ctx=Load()), args=[Name(id=**data**,

ctx=Load())], keywords=[])

Potential Insecure Deserialization vulnerabilities found in file: ../../additional\test_csrf.py,

Line 7: Call(func=Attribute(value=Name(id=**pickle**, ctx=Load()), attr=**loads**, ctx=Load()), args=[Name(id=**data**,

ctx=Load())], keywords=[])

Potential Insecure Deserialization vulnerabilities found in file: ../../additional\user.py,

Line 7: Call(func=Attribute(value=Name(id=**pickle**, ctx=Load()), attr=**loads**, ctx=Load()), args=[Name(id=**data**,

ctx=Load())], keywords=[])

Potential Insecure Deserialization vulnerabilities found in file: ../../additional\xss_detector_vulnerabilities.py,

Line 7: Call(func=Attribute(value=Name(id=**pickle**, ctx=Load()), attr=**loads**, ctx=Load()), args=[Name(id=**data**,

ctx=Load())], keywords=[])