File content analysis

Diamond inheritance problem

• BaseClass est une classe de base commune pour [**CorrectSubclass**, **SubclassWithoutSuper**]

• L**employé est une classe de base commune pour [**Développeur**,** Gestionnaire**]**

Not using the super() function

• La classe Employee hérite de [**ABC**] et n**utilise pas super() dans le constructeur.**

• La classe Developer hérite de [**Employee**] et n**utilise pas super() dans le constructeur.**

• La classe Manager hérite de [**Employee**] et n**utilise pas super() dans le constructeur.**

• La classe Mammifère hérite de [**Animal**] et n**utilise pas super() dans le constructeur.**

• La classe Primate hérite de [**Mammal**] et n**utilise pas super() dans le constructeur.**

• La classe Human hérite de [**Primate**] et n**utilise pas super() dans le constructeur.**

• La classe Person hérite de [**Human**] et n**utilise pas super() dans le constructeur.**

• La classe SubclassWithoutSuper hérite de [**BaseClass**] et n**utilise pas super() dans le constructeur.**

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

• La classe **DatabaseHandler** définit une variable statique **query**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « DatabaseHandler » définit un « curseur » de variable statique, qui peut provoquer des effets secondaires s**il est modifié d**une instance à l**autre.**

• La classe **DatabaseHandler** définit une variable statique **result**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « HermetyzacjaVisitor » définit une variable statique « has_public », qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « HermetyzacjaVisitor » définit une variable statique « tab_helper », qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « HermetyzacjaVisitor » définit une variable statique « has_private », qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **common_bases**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **inheritance_depth**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **base_classes**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « DependencyMapper » définit une variable statique « arbre », qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **base_methods**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **var_name**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « DependencyMapper » définit un « contenu » de variable statique, qui peut provoquer des effets secondaires s**il est modifié d**une instance à l**autre.**

• La classe **DependencyMapper** définit une variable statique **max_depth**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « DependencyMapper » définit une variable statique « bases », qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **new_depth**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe **DependencyMapper** définit une variable statique **derived_methods**, qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

• La classe « Utilisateur » définit une variable statique « shared_data », qui peut provoquer des effets secondaires si elle est modifiée d**une instance à l**autre.

Complex inheritance hierarchy

• La classe **Primate** a une hiérarchie d**héritage complexe avec une profondeur de 3, ce qui peut compliquer la**

base de code.

• La classe « Humain » a une hiérarchie d**héritage complexe avec une profondeur de 4, ce qui peut compliquer**

la base de code.

• La classe « Personne » a une hiérarchie d**héritage complexe avec une profondeur de 5, ce qui peut**

compliquer la base de code.

Missing abstract method implementations

• La classe « Développeur » ne met pas en œuvre toutes les méthodes abstraites de sa classe de base «

Employé ». Méthodes manquantes : signaler

• La classe « Manager » ne met pas en œuvre toutes les méthodes abstraites de sa classe de base « Employee

». Méthodes manquantes : travail

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=[])