BÁCH KHOA E-LEARNING

Đã bắt đầu vào lúc	Tuesday, 28 September 2021, 8:09 AM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Monday, 4 October 2021, 10:14 AM
Thời gian thực hiện	6 ngày 2 giờ
Điểm	4,00/4,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi **1**

Chính xác

Điểm 1,00 của 1,00

Let AST of a programming language be defined as follows:

class Program: #decl:List[Decl]

class Decl(ABC): #abstract class

class VarDecl(Decl): #name:str,typ:Type

class ConstDecl(Decl): #name:str,val:Lit

class Type(ABC): #abstract class

class IntType(Type)

class FloatType(Type)

class Lit(ABC): #abstract class

class IntLit(Lit): #val:int

and exception RedeclaredDeclaration:

class RedeclaredDeclaration(Exception): #name:str

Implement the methods of the following class Visitor to travel on the above ASST to detect redeclared declarations (throw exception RedeclaredDeclaration):

class StaticCheck(Visitor):

def visitProgram(self,ctx:Program,o:object): pass

def visitVarDecl(self,ctx:VarDecl,o:object):pass

def visitConstDecl(self,ctx:ConstDecl,o:object):pass

def visitIntType(self,ctx:IntType,o:object):pass

def visitFloatType(self,ctx:FloatType,o:object):pass

def visitIntLit(self,ctx:IntLit,o:object):pass

Your code starts at line 40

For example:

Test	Result
`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),VarDecl("a",FloatType())])`	`a`

Answer: (penalty regime: 0 %)

```
2
3   def visitProgram(self,ctx:Program,o:object):
4       o = []
5       for decl in ctx.decl:
6           o += [self.visit(decl,o)]
7
8   def visitVarDecl(self,ctx:VarDecl,o:object):
9       n = ctx.name
10      if n in o:
11          raise RedeclaredDeclaration(n)
12      return n
13
14  def visitConstDecl(self,ctx:ConstDecl,o:object):
15      n = ctx.name
16      if n in o:
17          raise RedeclaredDeclaration(n)
18      return n
19
20  def visitIntType(self,ctx:IntType,o:object):pass
21
22  def visitFloatType(self,ctx:FloatType,o:object):pass
23
24  def visitIntLit(self,ctx:IntLit,o:object):pass
```

	Test	Expected	Got	
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),VarDecl("a",FloatType())])`	a	a	✔
✔	`x = Program([VarDecl("b",IntType()),ConstDecl("b",IntLit(3)),VarDecl("a",FloatType())])`	b	b	✔
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("c",IntLit(3)),VarDecl("c",FloatType())])`	c	c	✔
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),VarDecl("c",FloatType())])`			✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **2**

Chính xác

Điểm 1,00 của 1,00

Let AST of a programming language be defined as follows:

class Program: #decl:List[Decl]

class Decl(ABC): #abstract class

class VarDecl(Decl): #name:str,typ:Type

class ConstDecl(Decl): #name:str,val:Lit

class Type(ABC): #abstract class

class IntType(Type)

class FloatType(Type)

class Lit(ABC): #abstract class

class IntLit(Lit): #val:int

and exceptions:

class RedeclaredVariable(Exception): #name:str

class RedeclaredConstant(Exception): #name:str

Implement the methods of the following class Visitor to travel on the above ASST to detect redeclared declarations (throw the exception corresponding to the second declaration with the same name):

class StaticCheck(Visitor):

    def visitProgram(self,ctx:Program,o:object): pass

    def visitVarDecl(self,ctx:VarDecl,o:object):pass

    def visitConstDecl(self,ctx:ConstDecl,o:object):pass

    def visitIntType(self,ctx:IntType,o:object):pass

    def visitFloatType(self,ctx:FloatType,o:object):pass

    def visitIntLit(self,ctx:IntLit,o:object):pass

Your code starts at line 45

For example:

Test	Result
`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),VarDecl("a",FloatType())])`	`Redeclared Varaible: a`

Answer: (penalty regime: 0 %)

```
2
3   def visitProgram(self,ctx:Program,o:object):
4       o = []
5       for decl in ctx.decl:
6           o += [self.visit(decl,o)]
7
8   def visitVarDecl(self,ctx:VarDecl,o:object):
9       n = ctx.name
10      if n in o:
11          raise RedeclaredVariable(n)
12      return n
13
14  def visitConstDecl(self,ctx:ConstDecl,o:object):
15      n = ctx.name
16      if n in o:
17          raise RedeclaredConstant(n)
18      return n
19
20  def visitIntType(self,ctx:IntType,o:object):pass
21
22  def visitFloatType(self,ctx:FloatType,o:object):pass
23
24  def visitIntLit(self,ctx:IntLit,o:object):pass
```

	Test	Expected	Got	
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),VarDecl("a",FloatType())])`	`Redeclared Varaible: a`	`Redeclared Varaible: a`	✔
✔	`x = Program([VarDecl("b",IntType()),ConstDecl("b",IntLit(3)),VarDecl("a",FloatType())])`	`Redeclared Constant: b`	`Redeclared Constant: b`	✔
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("c",IntLit(3)),VarDecl("c",FloatType())])`	`Redeclared Varaible: c`	`Redeclared Varaible: c`	✔
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),VarDecl("c",FloatType())])`			✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **3**

Chính xác

Điểm 1,00 của 1,00

Let AST of a programming language be defined as follows:

class Program: #decl:List[Decl]

class Decl(ABC): #abstract class

class VarDecl(Decl): #name:str,typ:Type

class ConstDecl(Decl): #name:str,val:Lit

class FuncDecl(Decl): #name:str,param:List[VarDecl],body:List[Decl]

class Type(ABC): #abstract class

class IntType(Type)

class FloatType(Type)

class Lit(ABC): #abstract class

class IntLit(Lit): #val:int

and exceptions:

class RedeclaredVariable(Exception): #name:str

class RedeclaredConstant(Exception): #name:str

class RedeclaredFunction(Exception): #name:str

Implement the methods of the following class Visitor to travel on the above AST to detect redeclared declarations (throw the exception corresponding to the second declaration with the same name) in the same scope:

class StaticCheck(Visitor):

    def visitProgram(self,ctx:Program,o:object): pass

    def visitVarDecl(self,ctx:VarDecl,o:object):pass

    def visitConstDecl(self,ctx:ConstDecl,o:object):pass

    def visitFuncDecl(self,ctx:FuncDecl,o:object):pass

    def visitIntType(self,ctx:IntType,o:object):pass

    def visitFloatType(self,ctx:FloatType,o:object):pass

    def visitIntLit(self,ctx:IntLit,o:object):pass

Your code starts at line 55

For example:

Test	Result
`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),FuncDecl("a",[],[])])`	`Redeclared Function: a`

Answer: (penalty regime: 0 %)

```
    def visitFuncDecl(self,ctx:FuncDecl,o:object):
        n=ctx.name
        listVar=ctx.param
        listbody=ctx.body
        name=[]
        if n in o:
            raise RedeclaredFunction(n)
        for nameVar in listVar:
            name+=[self.visit(nameVar,name)]
        for nameBody in listbody:
            name+=[self.visit(nameBody,name)]
        return n



    def visitIntType(self,ctx:IntType,o:object):pass

    def visitFloatType(self,ctx:FloatType,o:object):pass

    def visitIntLit(self,ctx:IntLit,o:object):pass
```

```
38      def visitIntLit(self,ctx:IntLit,o:Object):pass
39
40
41
```

	Test	Expected	Got	
✔	`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),FuncDecl("a",[],[])])`	`Redeclared Function: a`	`Redeclared Function: a`	✔
✔	`x = Program([VarDecl("b",IntType()),FuncDecl("a",[VarDecl("a",FloatType())], [ConstDecl("c",IntLit(3)),VarDecl("b",IntType()),VarDecl("c",IntType())])])`	`Redeclared Variable: c`	`Redeclared Variable: c`	✔
✔	`x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("m",FloatType())], [ConstDecl("c",IntLit(3)),VarDecl("d",IntType())])])`	`Redeclared Variable: m`	`Redeclared Variable: m`	✔
✔	`x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], [ConstDecl("c",IntLit(3)),VarDecl("d",IntType())])])`	`Redeclared Variable: d`	`Redeclared Variable: d`	✔
✔	`x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], [ConstDecl("c",IntLit(3)),FuncDecl("d",[],[])])])`	`Redeclared Function: d`	`Redeclared Function: d`	✔
✔	`x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], [ConstDecl("c",IntLit(3)),FuncDecl("foo",[VarDecl("x",IntType())], [VarDecl("x",IntType())])])])`	`Redeclared Variable: x`	`Redeclared Variable: x`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **4**

Chính xác

Điểm 1,00 của 1,00

Let AST of a programming language be defined as follows:

class Program: #decl:List[Decl]

class Decl(ABC): #abstract class

class VarDecl(Decl): #name:str,typ:Type

class ConstDecl(Decl): #name:str,val:Lit

class FuncDecl(Decl): #name:str,param:List[VarDecl],body:Tuple(List[Decl],List[Expr])

class Type(ABC): #abstract class

class IntType(Type)

class FloatType(Type)

class Expr(ABC): #abstract class

class Lit(Expr): #abstract class

class IntLit(Lit): #val:int

class Id(Expr): #name:str

and exceptions:

class RedeclaredVariable(Exception): #name:str

class RedeclaredConstant(Exception): #name:str

class RedeclaredFunction(Exception): #name:str

class UndeclaredIdentifier(Exception): #name:str

Implement the methods of the following class Visitor to travel on the above AST to detect undeclared declarations (throw the exception UndeclaredIdentifier). Note that the redeclared declarations exception also is thrown if a redeclared declaration is detected:

class StaticCheck(Visitor):

def visitProgram(self,ctx:Program,o:object): pass

def visitVarDecl(self,ctx:VarDecl,o:object):pass

def visitConstDecl(self,ctx:ConstDecl,o:object):pass

def visitFuncDecl(self,ctx:FuncDecl,o:object):pass

def visitIntType(self,ctx:IntType,o:object):pass

def visitFloatType(self,ctx:FloatType,o:object):pass

def visitIntLit(self,ctx:IntLit,o:object):pass

def visitId(self,ctx:Id,o:object):pass

Your code starts at line 65

For example:

Test	Result
`x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),FuncDecl("a",[],([],[]))])`	`Redeclared Function: a`

Answer: (penalty regime: 0 %)

```
            name[0]+=[self.visit(nameVar,name)]
        for nameBody in listbody:
            name[0]+=[self.visit(nameBody,name+o)]
        for id in listId:
            self.visit(id,name+o)
```

```
39      def visitIntType(self,ctx:IntType,o:object):pass
40
41      def visitFloatType(self,ctx:FloatType,o:object):pass
42
43      def visitIntLit(self,ctx:IntLit,o:object):pass
44
45      def visitId(self,ctx:Id,o:object):
46          n = ctx.name
47          for a in o:
48              if n in a:
49                  return True
50          raise UndeclaredIdentifier(n)
51
52
```


✔	x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("m",FloatType())], ([ConstDecl("c",IntLit(3)),VarDecl("d",IntType())],[]))])	Redeclared Variable: m	Redeclared Variable: m	✔
✔	x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], ([ConstDecl("c",IntLit(3)),VarDecl("d",IntType())],[]))])	Redeclared Variable: d	Redeclared Variable: d	✔
✔	x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], ([ConstDecl("c",IntLit(3)),FuncDecl("d",[],([],[]))],[]))])	Redeclared Function: d	Redeclared Function: d	✔
✔	x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], ([ConstDecl("c",IntLit(3)),FuncDecl("foo",[VarDecl("x",IntType())], ([VarDecl("x",IntType())],[]))],[]))])	Redeclared Variable: x	Redeclared Variable: x	✔
✔	x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], ([ConstDecl("c",IntLit(3)),FuncDecl("foo",[VarDecl("x",IntType())], ([VarDecl("y",IntType()),VarDecl("z",IntType())], [Id("y"),Id("x"),Id("foo"),Id("c"),Id("m"),Id("a")]))], [Id("foo"),Id("d"),Id("z")]))])	Undeclared Identifier: z	Undeclared Identifier: z	✔
✔	x = Program([VarDecl("a",IntType()),ConstDecl("b",IntLit(3)),FuncDecl("c",[], ([],[IntLit(1),Id("a"),Id("d"),Id("b")]))])	Undeclared Identifier: d	Undeclared Identifier: d	✔
✔	x = Program([VarDecl("b",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("n",FloatType())], ([ConstDecl("c",IntLit(3)),VarDecl("d",IntType())], [Id("a"),Id("b"),Id("c"),Id("d"),IntLit(3),Id("m"),Id("q"),Id("n")]))])	Undeclared Identifier: q	Undeclared Identifier: q	✔
✔	x = Program([VarDecl("t",IntType()),FuncDecl("a", [VarDecl("m",FloatType()),VarDecl("b",IntType()),VarDecl("d",FloatType())], ([ConstDecl("c",IntLit(3)),FuncDecl("foo",[VarDecl("x",IntType())], ([VarDecl("y",IntType()),VarDecl("z",IntType())], [Id("y"),Id("x"),Id("foo"),Id("c"),Id("m"),Id("a"),Id("t")])), FuncDecl("foo1",[],([],[Id("foo"),Id("d"),Id("x")]))], [Id("foo"),Id("d"),Id("foo1")]))])	Undeclared Identifier: x	Undeclared Identifier: x	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

◄ Name Quiz

Chuyển tới...

Link Video của buổi 28/9/2021 ►