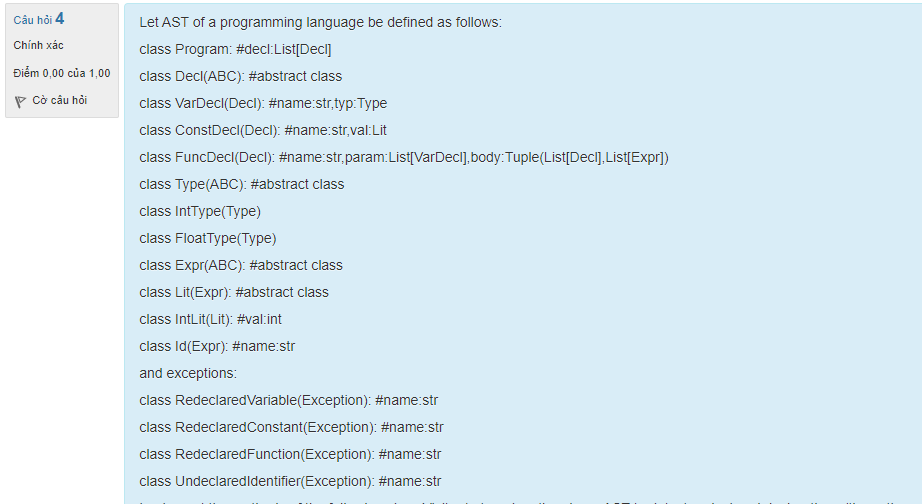
Bài 4



# from functools import reduce

class StaticCheck(Visitor):

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

o = {

'parent': [],

'child': []

}

for x in ctx.decl:

self.visit(x, o)

pass

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

if ([\*filter(lambda x:ctx.name == x, o["child"])]):

raise RedeclaredVariable(ctx.name)

o["child"].append(ctx.name)

pass

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

if ([\*filter(lambda x:ctx.name == x, o["child"])]):

raise RedeclaredConstant(ctx.name)

o["child"].append(ctx.name)

pass

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

# check only for 1 scope

if ([\*filter(lambda x:ctx.name == x, o["child"])]):

raise RedeclaredFunction(ctx.name)

o["child"].append(ctx.name)

#print("func name", ctx.name)

#print("Object", o)

func\_o = {

"parent": o["child"],

"child": []

}

# visit para

for x in ctx.param:

self.visit(x, func\_o)

# visit body\_decl

body\_decl = ctx.body[0]

#print(body\_decl)

for x in body\_decl:

self.visit(x, func\_o)

# visit body\_expr

body\_expr = ctx.body[1]

for x in body\_expr:

self.visit(x, o["parent"] + o["child"] + func\_o["child"])

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):

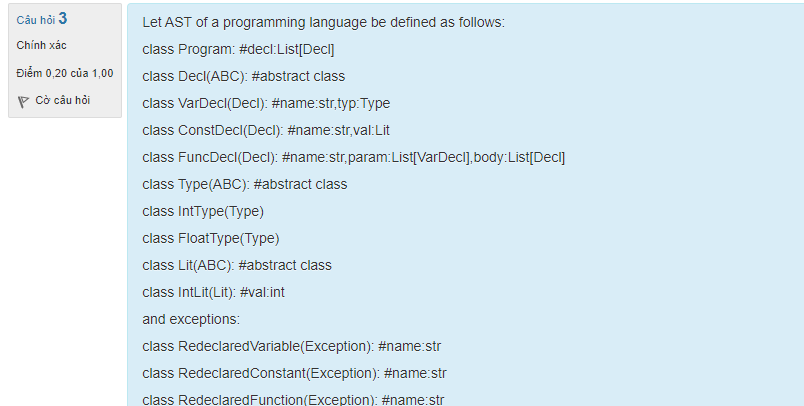
#print("O", o)

#print("Current id", ctx.name)

#print("Undeclared ?", len([x for x in o if (ctx.name == x)]) == 0)

if (len([x for x in o if (ctx.name == x)]) == 0):

raise UndeclaredIdentifier(ctx.name)



class StaticCheck(Visitor):

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

o = []

for x in ctx.decl:

self.visit(x, o)

o += x.name

pass

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

if (ctx.name in o):

raise RedeclaredVariable(ctx.name)

pass

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

if (ctx.name in o):

raise RedeclaredConstant(ctx.name)

pass

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

# check only for 1 scope

if (ctx.name in o):

raise RedeclaredFunction(ctx.name)

func\_o = []

# visit para

for x in ctx.param:

self.visit(x, func\_o)

func\_o += x.name

# visit body\_decl

body\_decl = ctx.body

for x in body\_decl:

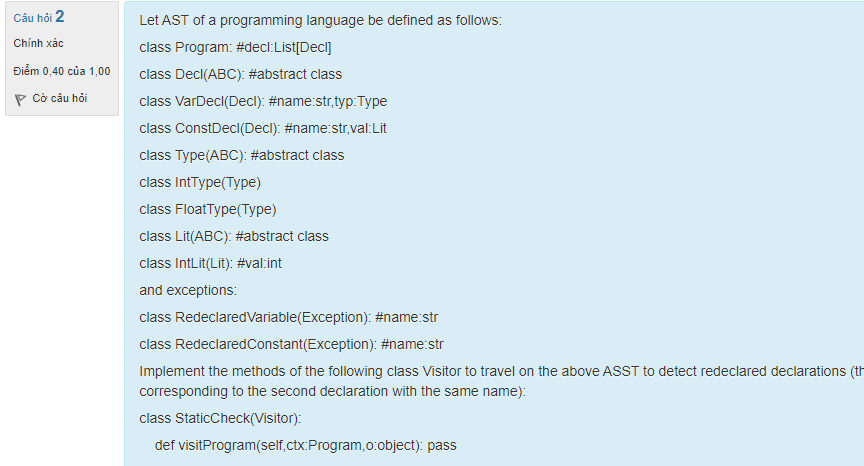
self.visit(x, func\_o)

func\_o += x.name

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

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

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



class StaticCheck(Visitor):

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

o = []

for x in ctx.decl:

self.visit(x, o)

o += x.name

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

if (ctx.name in o):

raise RedeclaredDeclaration("Redeclared Varaible: "+ctx.name)

return ctx

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

if (ctx.name in o):

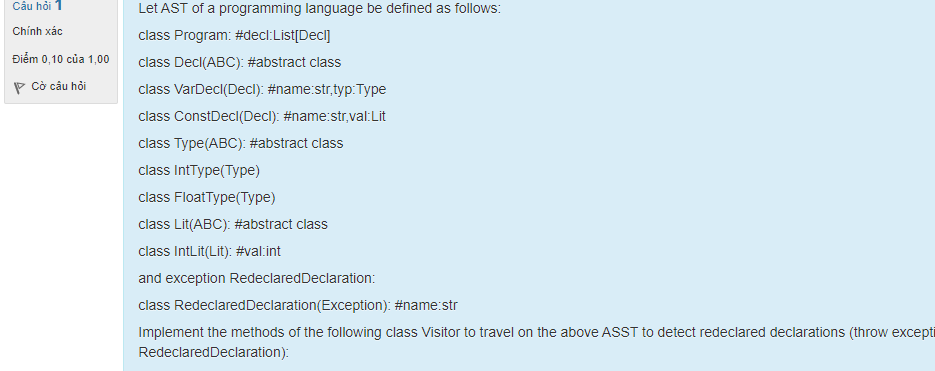
raise RedeclaredDeclaration("Redeclared Constant: " + ctx.name)

return ctx

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

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

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



class StaticCheck(Visitor):

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

o = []

for x in ctx.decl:

if (x.name in o):

raise RedeclaredDeclaration(x.name)

o += x.name

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

if (ctx.name in o):

raise RedeclaredDeclaration(ctx.name)

return ctx

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

if (ctx.name in o):

raise RedeclaredDeclaration(ctx.name)

return ctx

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

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

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