Software Quality Assurance
Assignment 2

Submitted by :
Shiv Kumar
2016UIT2563
Semester 8
IT2

Practical : Write a program to determine the values of CK metric suite for each class.

main.py

```
import sys
import Raw
from CK import *
try:
   from StringIO import StringIO
except ImportError:
   from io import StringIO
import os

def get_complexity_number(snippet, strio, max=0):
   """Get the complexity number from the printed string."""
   # Report from the lowest complexity number.
   get_code_complexity(snippet, max)
   strio_val = strio.getvalue()
   strio.close()
   if strio_val:
      return int(strio_val.split()[-1].strip("()"))
   else:
      return None

def collectingPath(direc, pathList = []):
   count = 0
   for filename in os.listdir(direc):
      pathway = os.path.join(direc, filename)
      if os.path.isfile(pathway) and pathway.endswith(".py"):
         pathList.append(pathway)
      elif os.path.isdir(pathway):
         pathList = collectingPath(pathway,  pathList)
   #print(len(pathList))
   return pathList

def read_files(path_list):
   mdef = ""
   for path in path_list:
      with open(path,encoding="utf8", mode = 'r') as reader:
         val = reader.read()
         mdef += "\n"
         mdef += val

   return mdef

path_List =collectingPath("/Users/championballer/Documents/Code/Labs/Software Quality
Assurance/Baselines/ddpg")
mdef = read_files(path_List)
loca = Raw.analyze(mdef)
comment_percentage = loca.comments/(loca.loc-loca.blank-loca.comments)
print("Overall Report")
print("LOC: ",loca.loc)
print("Multi Line of Comment: ", loca.multi)
print("Single Line of Comment: ", loca.comments)
print("Comment Percentage: ",comment_percentage)
large_Class_Method(mdef)
mdef = remove_comments_and_docstrings(mdef)
print(mdef)
```

```
inherit_tree, all_node, astree = inheritance_tree(mdef)
print(all_node)
CK_MOOD_Metrics(mdef, inherit_tree, all_node, astree)
```

CK.py

```
import ast
import copy
import re, sys
from McCabe import get_code_complexity
try:
    from StringIO import StringIO
except ImportError:
    from io import StringIO
import textwrap
import pandas as pd
import numpy as np
import tokenize
from Raw import *

f = open("result5.txt",'w')

mdef = '''
class A(object):
    def meth(self):
        return sum(i for i in range(10) if i - 2 < 5)
    def fib(self, n):
        pass
class B(A):
    def _thi(self, mo):
        for i in range(5):
            for j in i:
                f.write(j)
        return sum(i for i in range(10) if i - 2 < 5)
    def fib(self, n):
        self.p = 0
        g = A()
        g.meth()
        g.fib(n)
        h=0
        __d = h
        f, k = 0
        if n < 2: return 1
        return fib(n - 1) + fib(n - 2)
class C(A, B):
    def fr(self):
        return 34

class E(object):
    def fr(self):
        return 34
class F(E):
    def meth(self):
        return sum(i for i in range(10) if i - 2 < 5)
    def fib(self, n):
        pass
    '''
def findAllMethods(script):
    reg = re.compile('((?:^[ \t]*)def \w+\(.*\): *(?=.*?[^ \t\n]).*\r?\n)'
                     '|'
```

```
            '((^[ \t]*)def \w+\(.*\): *\r?\n'
            '(?:[ \t]*\r?\n)*'
            '\\3([ \t]+)[^ \t].*\r?\n'
            '(?:[ \t]*\r?\n)*'
            '(\\3\\4.*\r?\n(?: *\r?\n)*)*)',
            re.MULTILINE)
    arr =[]

    for ma in  reg.finditer(script):
        #f.write ("aaaa ",ma.group())
        arr.append(ma.group())

    return arr

def get_complexity_number(snippet, strio, max=0):
    """Get the complexity number from the f.writeed string."""
    # Report from the lowest complexity number.
    get_code_complexity(snippet, max)
    strio_val = strio.getvalue()
    strio.close()
    if strio_val:
        return int(strio_val.split()[-1].strip("()"))
    else:
        return None

def weighted_method_per_class(source):
    f.write("\n")
    f.write("Metrics Name: Weighted Method Per Class\n")
    classes = re.findall('(class[\s\S]*?)(?=class|$)',source)

    for clas in classes:
        #print(clas)
        astt = ast.parse(clas)
        def_class = [n for n in ast.walk(astt) if type(n) == ast.ClassDef]
        f.write("\n")
        f.write("    Class Name: ")
        f.write(def_class[0].name)
        f.write("\n")
        methods = findAllMethods(clas)
        wmpc=0
        for method in methods:
            method = textwrap.dedent(method)
            a = ast.parse(method)
            definitions = [n for n in ast.walk(a) if type(n) == ast.FunctionDef]
            #print(method)
            stdout = sys.stdout
            strio = StringIO()
            sys.stdout = strio

            #print(findAllMethods(method))
            val = get_complexity_number(method, strio)
            sys.stdout = stdout
            f.write("           Method Name: ")
            f.write(definitions[0].name)
            f.write("\n           Complexity: ")
            f.write(str(val))
            #print()
            wmpc+=val
        #print("       Weighted Method Per Class for ",def_class[0].name, "is ", wmpc)
        f.write("       Weighted Method Per Class for ")
```

```
        f.write(def_class[0].name)
        f.write("is ")
        f.write(str(wmpc))
        #print(wmpc)
        f.write("\n")
def inheritance_tree(source):
    f.write("Generating AST.....\n")
    a = ast.parse(source)
    all_node = set()
    definitions = [n for n in ast.walk(a) if type(n) == ast.ClassDef]
    #f.write(definitions)
    inheritance_tree = {}
    for i in definitions:
        all_node.update([i.name])
        print(i.name)
        inheritance_tree[i.name] = []
        f.write("\nParent Class: ")
        f.write(i.name)
        f.write("\n")
        for j in i.bases:
            try:
                if not j.id== "object":
                    inheritance_tree[i.name].append(j.id)
                    f.write("\n    Inherited Class ")
                    f.write(j.id)
                    f.write("\n")
            except:
                pass
    print(all_node)
    return inheritance_tree, all_node, definitions

def depth_of_inheritance_tree_util(tree,counter, max_counter):
    f.write("\n")
    f.write("Metrics Name: Depth of Inheritance Tree: \n")
    for child in tree:
        counter+=1
        max_counter = depth_of_inheritance_tree(tree, child, counter, max_counter)
        max_counter = max(max_counter, counter)
        counter-=1
        #print(child, max_counter)
    f.write("\n    DIT: ")
    f.write(str(max_counter))
    f.write("\n")
    return max_counter
def depth_of_inheritance_tree(tree, node, counter, max_counter):
    clas = list(tree.keys())
    if node in clas:
        for child in tree.get(node, None):
            if not child == None and not child == "":
                counter+=1

                max_counter = depth_of_inheritance_tree(tree, child, counter, max_counter)
                max_counter = max(max_counter, counter)
                #print(max_counter, node)
                counter-=1
    return max_counter

def Number_of_child(tree, all_node):
    f.write("\n")
    f.write("Metrics Name: Number of Child")
```

```
    f.write("\n")
    child = {}
    for i in all_node:
        child[i] = []

    for node in all_node:
        for parent in tree:
            if node in tree[parent]:
                child[node].append(parent)
    for parent in child:
        f.write("\n    Class: ")
        f.write(parent)
        f.write("\n        Number of Child: ")
        f.write(str(len(child[parent])))
        f.write("\n")
    return child

def attr_hiding_factor(astree):
    f.write("\n")
    f.write("Metrics Name: Attribute Hiding Factor")
    f.write("\n")
    for class_obj in astree:
        #f.write("Class Name: ", class_obj.name)
        variable_count = 0
        private_var_count = 0
        class_attr= set()
        class_pr_attr = set()
        for func_obj in class_obj.body:
            try:
                for statements in func_obj.body:
                    if isinstance(statements,ast.Assign):
                        for variables in statements.targets:
                            if isinstance(variables,ast.Tuple):
                                if isinstance(variables, ast.Attribute):
                                    if variables.attr[0] == "_":
                                        class_attr.update(variables.attr)
                                        class_pr_attr.update(variables.attr)
                                    else:
                                        class_attr.update(variables.attr)
                                else:
                                    for var in variables.elts:
                                        #f.write(var.id)
                                        variable_count+=1
                                        if var.id[0] == "_":
                                            private_var_count+=1
                            elif isinstance(variables,ast.Name):
                                #f.write(variables.id)
                                variable_count+=1
                                if variables.id[0] == "_":
                                    private_var_count+=1
                            elif isinstance(variables, ast.Attribute):
                                if variables.attr[0] == "_":
                                    class_attr.update(variables.attr)
                                    class_pr_attr.update(variables.attr)
                                else:
                                    class_attr.update(variables.attr)
            except:
                pass
            variable_count +=len(class_attr)
            private_var_count +=len(class_pr_attr)
```

```
        if variable_count>0:
            hiding_factor = private_var_count/variable_count
        else:
            hiding_factor = 0
        f.write("\n    Class: "+class_obj.name+": ")
        f.write("\n        FHF: "+ str(hiding_factor))

def remove_comments_and_docstrings(source):
    """
    Returns 'source' minus comments and docstrings.
    """
    io_obj = StringIO(source)
    out = ""
    prev_toktype = tokenize.INDENT
    last_lineno = -1
    last_col = 0
    for tok in tokenize.generate_tokens(io_obj.readline):
        token_type = tok[0]
        token_string = tok[1]
        start_line, start_col = tok[2]
        end_line, end_col = tok[3]
        ltext = tok[4]
        # The following two conditionals preserve indentation.
        # This is necessary because we're not using tokenize.untokenize()
        # (because it spits out code with copious amounts of oddly-placed
        # whitespace).
        if start_line > last_lineno:
            last_col = 0
        if start_col > last_col:
            out += (" " * (start_col - last_col))
        # Remove comments:
        if token_type == tokenize.COMMENT:
            pass
        # This series of conditionals removes docstrings:
        elif token_type == tokenize.STRING:
            if prev_toktype != tokenize.INDENT:
        # This is likely a docstring; double-check we're not inside an operator:
                if prev_toktype != tokenize.NEWLINE:
                    # Note regarding NEWLINE vs NL: The tokenize module
                    # differentiates between newlines that start a new statement
                    # and newlines inside of operators such as parens, brackes,
                    # and curly braces.  Newlines inside of operators are
                    # NEWLINE and newlines that start new code are NL.
                    # Catch whole-module docstrings:
                    if start_col > 0:
                        # Unlabelled indentation means we're inside an operator
                        out += token_string
                    # Note regarding the INDENT token: The tokenize module does
                    # not label indentation inside of an operator (parens,
                    # brackets, and curly braces) as actual indentation.
                    # For example:
                    # def foo():
                    #     "The spaces before this docstring are tokenize.INDENT"
                    #     test = [
                    #         "The spaces before this string do not get a token"
                    #     ]
        else:
            out += token_string
        prev_toktype = token_type
        last_col = end_col
```

```
        last_lineno = end_line
    return out

def method_hiding_factor(astree):
    f.write("\n")
    f.write("Metrics Name: Method Hiding Factor")
    f.write("\n")
    for class_obj in astree:
        #f.write("Class Name: ", class_obj.name)
        function_count = 0
        private_func_count = 0
        for func_obj in class_obj.body:
            function_count += 1
            if isinstance(func_obj, ast.FunctionDef):
                if func_obj.name[0] == "_":
                    private_func_count += 1
        hiding_factor = private_func_count/function_count
        f.write("\n    Class "+class_obj.name+": ")
        f.write("\n        MHF: "+str(hiding_factor))

def BFS(child_tree, inheritance_tree,  start, visited, all_inherit):
    for child in child_tree[start]:
        if not child in visited:
            visited.append(child)
            for parent in inheritance_tree[child]:
                all_inherit[child]+=all_inherit[parent]
            #all_inherit[child]+=all_inherit[start]
            BFS(child_tree, inheritance_tree,  child, visited, all_inherit)
        else:
            return


def method_inheritance_factor(inheritance_tree, child_tree, astree, all_node):
    f.write("\n")
    f.write("Metrics Name: Method Inheritance Factor")
    f.write("\n")
    class_to_method = {}
    for classes in astree:
        all_method = []
        for methods in classes.body:
            if isinstance(methods, ast.FunctionDef):
                all_method.append(methods.name)
        class_to_method[classes.name] = all_method.copy()
    class_inherit_method = copy.deepcopy(class_to_method)
    visited = []
    for node in inheritance_tree:
        if len(inheritance_tree[node]) == 0 and node in all_node:
            BFS(child_tree, inheritance_tree, node, visited, class_inherit_method)

    #f.write(class_inherit_method)
    #f.write( class_to_method)
    inherit_count = {}
    for node in class_to_method:
        inherit_count[node] = len(set(class_inherit_method[node])-set(class_to_method[node]))
    #f.write(inherit_count)
    MIF = {}
    for node in class_to_method:
        if not inherit_count[node] == 0:
            MIF[node] =  inherit_count[node]/len(set(class_inherit_method[node]))
        else:
```

```
            MIF[node] = 0
    #f.write(inherit_count)
    for factor in MIF:
        f.write("\n    Class: "+str(factor)+"\n")
        f.write("        MIF: "+str(MIF[factor])+"\n")


def coupling_factor(astree):
    f.write("\n")
    f.write("Metrics Name: Coupling Factor")
    f.write("\n")
    classes = []
    for clas in astree:
        classes.append(clas.name)
    #f.write(classes)
    #df = pd.DataFrame(np.zeros(len(classes), len(classes)))
    df = pd.DataFrame(0, index=range(len(classes)), columns=range(len(classes)))
    #f.write(df)
    df.columns = classes
    df.index = classes
    #f.write(df)
    coupling = 0
    for clas in astree:
        for line in clas.body:
            if isinstance(line, ast.Assign) and isinstance(line.value, ast.Call):
                if isinstance(line.value.func, ast.Name):
                    if line.value.func.id in classes:
                        coupling+=1
                        print("Found", line.value.func.id)
                        df[clas.name][line.value.func.id] = 1
    print(df.to_string())

    for clas in astree:
        for funcs in clas.body:
            if isinstance(funcs, ast.FunctionDef):
                for line in funcs.body:
                    if isinstance(line, ast.Assign) and isinstance(line.value, ast.Call):
                        if isinstance(line.value.func, ast.Name):
                            if line.value.func.id in classes:
                                coupling+=1
                                print("Found")
                                df[clas.name][line.value.func.id] = 1
    f.write("Coupling Between Objects: \n")
    f.write(df.to_string())
    f.write("\n    COF: "+str(coupling/len(classes))+str("\n"))

def large_Class_Method(source):
    classes = re.findall('(class[\s\S]*?)(?=class|$)',source)
    for clas in classes:
        method_list = findAllMethods(clas)
        clas_len = 0
        clas_multi = 0
        clas_comments = 0
        c = ast.parse(clas)
        class_def = [n for n in ast.walk(c) if type(n) == ast.ClassDef]
        print("Class Name: ", class_def[0].name)
        print("  Method Number: ", len(method_list))
        for meth in method_list:
            meth = textwrap.dedent(meth)
            a = ast.parse(meth)
```

```
            definitions = [n for n in ast.walk(a) if type(n) == ast.FunctionDef]
            print("  Method Name: ", definitions[0].name)
            loc = analyze(meth)
            print("    Parameter Length: ", len(definitions[0].args.args)-1)
            comment_percentage = loc.comments/(loc.loc-loc.blank-loc.comments)
            #print("McCabe Cyclomatric Complexity: ", val)
            print("    LOC: ",loc.loc)
            clas_len+=loc.loc
            print("    Multi Line of Comment: ", loc.multi)
            clas_multi+=loc.multi
            print("    Single Line of Comment: ", loc.comments)
            clas_comments += loc.comments
            print("    Comment Percentage: ",comment_percentage)
            #print(clas)

        print("LOC: ",clas_len)
        print("Multi Line of Comment: ", clas_multi)
        print("Single Line of Comment: ", clas_comments)
        print()

def CK_MOOD_Metrics(mdef, inheritance_tree, all_node, astree):
    maxi = depth_of_inheritance_tree_util(inheritance_tree, 0, 0)
    child_tree = Number_of_child(inheritance_tree, all_node)
    #f.write(child_tree)
    attr_hiding_factor(astree)
    method_hiding_factor(astree)
    method_inheritance_factor(inheritance_tree, child_tree, astree, all_node)
    weighted_method_per_class(mdef)
    coupling_factor(astree)
    #large_Class_Method(mdef)
if __name__ == "__main__":

    inheritance_tree, all_node, astree = inheritance_tree(mdef)
    CK_MOOD_Metrics(mdef, inheritance_tree, all_node, astree)
    loc = Raw.analyze(mdef)
    comment_percentage = loc.comments/(loc.loc-loc.blank-loc.comments)
    print("Overall Report: ")
    #print("McCabe Cyclomatric Complexity: ", val)
    print("LOC: ",loc.loc)
    print("Multi Line of Comment: ", loc.multi)
    print("Single Line of Comment: ", loc.comments)
    print("Comment Percentage: ",comment_percentage)
    f.close()
```

Result file :

Parent Class: Model

Parent Class: Actor

    Inherited Class Model

Parent Class: Critic

    Inherited Class Model

Parent Class: RingBuffer

Parent Class: Memory

Parent Class: DDPG

Parent Class: AdaptiveParamNoiseSpec

Parent Class: ActionNoise

Parent Class: NormalActionNoise

Inherited Class ActionNoise

Parent Class: OrnsteinUhlenbeckActionNoise

Inherited Class ActionNoise

Metrics Name: Depth of Inheritance Tree:

DIT: 2

Metrics Name: Number of Child

Class: Critic
Number of Child: 0

Class: Model
Number of Child: 2

Class: DDPG
Number of Child: 0

Class: Actor
Number of Child: 0

Class: NormalActionNoise
Number of Child: 0

Class: ActionNoise
Number of Child: 2

Class: RingBuffer
Number of Child: 0

Class: AdaptiveParamNoiseSpec
Number of Child: 0

Class: OrnsteinUhlenbeckActionNoise
Number of Child: 0

Class: Memory
Number of Child: 0

Metrics Name: Attribute Hiding Factor

Class: Model:
FHF: 0.0
Class: Actor:
FHF: 0.0
Class: Critic:
FHF: 0.0
Class: RingBuffer:
FHF: 0.0

Class: Memory:
FHF: 0.0
Class: DDPG:
FHF: 0.0
Class: AdaptiveParamNoiseSpec:
FHF: 0.0
Class: ActionNoise:
FHF: 0
Class: NormalActionNoise:
FHF: 0.0
Class: OrnsteinUhlenbeckActionNoise:
FHF: 0.0
Metrics Name: Method Hiding Factor

Class Model:
MHF: 0.25
Class Actor:
MHF: 1.0
Class Critic:
MHF: 0.6666666666666666
Class RingBuffer:
MHF: 0.6
Class Memory:
MHF: 0.25
Class DDPG:
MHF: 0.06666666666666667
Class AdaptiveParamNoiseSpec:
MHF: 0.5
Class ActionNoise:
MHF: 0.0
Class NormalActionNoise:
MHF: 1.0
Class OrnsteinUhlenbeckActionNoise:
MHF: 0.75
Metrics Name: Method Inheritance Factor

Class: Model
MIF: 0

Class: Actor
MIF: 0.6

Class: Critic
MIF: 0.5

Class: RingBuffer
MIF: 0

Class: Memory
MIF: 0

Class: DDPG
MIF: 0

Class: AdaptiveParamNoiseSpec
MIF: 0

Class: ActionNoise
MIF: 0

Class: NormalActionNoise
MIF: 0.25

Class: OrnsteinUhlenbeckActionNoise
MIF: 0

Metrics Name: Weighted Method Per Class

Class Name: Model
Method Name: __init__
Complexity: 1 Method Name: vars
Complexity: 1 Method Name: trainable_vars
Complexity: 1 Method Name: perturbable_vars
Complexity: 1 Weighted Method Per Class for Modelis 4

Class Name: Actor
Method Name: __init__
Complexity: 1 Method Name: __call__
Complexity: 1 Weighted Method Per Class for Actoris 2

Class Name: Critic
Method Name: __init__
Complexity: 1 Method Name: __call__
Complexity: 1 Method Name: output_vars
Complexity: 1 Weighted Method Per Class for Criticis 3

Class Name: RingBuffer
Method Name: __init__
Complexity: 1 Method Name: __len__
Complexity: 1 Method Name: __getitem__
Complexity: 2 Method Name: get_batch
Complexity: 1 Method Name: append
Complexity: 3 Method Name: array_min2d
Complexity: 2 Weighted Method Per Class for RingBufferis 10

Class Name: Memory
Method Name: __init__
Complexity: 1 Method Name: sample
Complexity: 1 Method Name: append
Complexity: 2 Method Name: nb_entries
Complexity: 1 Method Name: normalize
Complexity: 2 Method Name: denormalize
Complexity: 2 Method Name: reduce_std
Complexity: 1 Method Name: reduce_var
Complexity: 1 Method Name: get_target_updates
Complexity: 2 Method Name: get_perturbed_actor_updates
Complexity: 3 Weighted Method Per Class for Memoryis 16

Class Name: DDPG
Method Name: setup_target_network_updates
Complexity: 1 Method Name: setup_param_noise
Complexity: 1 Method Name: setup_actor_optimizer
Complexity: 1 Method Name: setup_critic_optimizer
Complexity: 3 Method Name: setup_popart
Complexity: 2 Method Name: setup_stats
Complexity: 4 Method Name: step
Complexity: 4 Method Name: store_transition
Complexity: 3 Method Name: train
Complexity: 2 Method Name: initialize
Complexity: 1 Method Name: update_target_net

Complexity: 1 Method Name: get_stats
Complexity: 3 Method Name: adapt_param_noise
Complexity: 5 Method Name: reset
Complexity: 3 Method Name: _run
Complexity: 1 Method Name: test_popart
Complexity: 1 Method Name: test_noise_normal
Complexity: 1 Method Name: test_noise_ou
Complexity: 1 Method Name: test_noise_adaptive
Complexity: 1 Weighted Method Per Class for DDPGis 39

Class Name: AdaptiveParamNoiseSpec
Method Name: __init__
Complexity: 1 Method Name: adapt
Complexity: 2 Method Name: get_stats
Complexity: 1 Method Name: __repr__
Complexity: 1 Weighted Method Per Class for AdaptiveParamNoiseSpecis 5

Class Name: ActionNoise
Method Name: reset
Complexity: 1 Weighted Method Per Class for ActionNoiseis 1

Class Name: NormalActionNoise
Method Name: __init__
Complexity: 1 Method Name: __call__
Complexity: 1 Method Name: __repr__
Complexity: 1 Weighted Method Per Class for NormalActionNoiseis 3

Class Name: OrnsteinUhlenbeckActionNoise
Method Name: __init__
Complexity: 1 Method Name: __call__
Complexity: 1 Method Name: reset
Complexity: 1 Method Name: __repr__
Complexity: 1 Method Name: as_scalar
Complexity: 3 Weighted Method Per Class for OrnsteinUhlenbeckActionNoiseis 7

Metrics Name: Coupling Factor
Coupling Between Objects:

	Model	Actor	Critic	RingBuffer	Memory	DDPG	AdaptiveParamNoiseSpec	ActionNoise	NormalActionNoise	OrnsteinUhlenbeckActionNoise
Model	0	0	0	0	0	0	0	0	0	0
Actor	0	0	0	0	0	0	0	0	0	0
Critic	0	0	0	0	0	0	0	0	0	0
RingBuffer	0	0	0	0	1	0	0	0	0	0
Memory	0	0	0	0	0	0	0	0	0	0
DDPG	0	0	0	0	0	0	0	0	0	0
AdaptiveParamNoiseSpec	0	0	0	0	0	0	0	0	0	0
ActionNoise	0	0	0	0	0	0	0	0	0	0
NormalActionNoise	0	0	0	0	0	0	0	0	0	0
OrnsteinUhlenbeckActionNoise	0	0	0	0	0	0	0	0	0	0

COF: 0.5

Console Output :

```
(base) Shivs-Air:Traditional-CK-MOOD-Metrics-For-Python championballer$ python3 main.py
Overall Report
LOC:  895
Multi Line of Comment:  0
Single Line of Comment:  62
Comment Percentage:  0.09198813056379822
Class Name:  Model
  Method Number:  4
  Method Name:  __init__
    Parameter Length:  2
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  vars
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  trainable_vars
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  perturbable_vars
    Parameter Length:  0
```

```
  Method Name:  perturbable_vars
    Parameter Length:  0
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  14
Multi Line of Comment:  0
Single Line of Comment:  0

Class Name:  Actor
  Method Number:  2
  Method Name:  __init__
    Parameter Length:  3
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __call__
    Parameter Length:  2
    LOC:  8
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  12
Multi Line of Comment:  0
Single Line of Comment:  0

Class Name:  Critic
  Method Number:  3
  Method Name:  __init__
    Parameter Length:  2
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __call__
    Parameter Length:  3
    LOC:  7
    Multi Line of Comment:  0
    Single Line of Comment:  1
    Comment Percentage:  0.2
  Method Name:  output_vars
    Parameter Length:  0
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  15
Multi Line of Comment:  0
Single Line of Comment:  1

Class Name:  RingBuffer
```

```
Class Name:  RingBuffer
  Method Number:  6
  Method Name:  __init__
    Parameter Length:  3
    LOC:  6
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __len__
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __getitem__
    Parameter Length:  1
    LOC:  5
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  get_batch
    Parameter Length:  1
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  append
    Parameter Length:  1
    LOC:  13
    Multi Line of Comment:  0
    Single Line of Comment:  3
    Comment Percentage:  0.375
  Method Name:  array_min2d
    Parameter Length:  0
    LOC:  7
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  37
Multi Line of Comment:  0
Single Line of Comment:  3

Class Name:  Memory
  Method Number:  10
  Method Name:  __init__
    Parameter Length:  3
    LOC:  9
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  sample
    Parameter Length:  1
    LOC:  19
```

```
Method Name:  sample
  Parameter Length:  1
  LOC:  19
  Multi Line of Comment:  0
  Single Line of Comment:  1
  Comment Percentage:  0.0666666666666667
Method Name:  append
  Parameter Length:  6
  LOC:  10
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  nb_entries
  Parameter Length:  0
  LOC:  4
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  normalize
  Parameter Length:  1
  LOC:  6
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  denormalize
  Parameter Length:  1
  LOC:  5
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  reduce_std
  Parameter Length:  2
  LOC:  3
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  reduce_var
  Parameter Length:  2
  LOC:  5
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  get_target_updates
  Parameter Length:  2
  LOC:  14
  Multi Line of Comment:  0
  Single Line of Comment:  0
  Comment Percentage:  0.0
Method Name:  get_perturbed_actor_updates
  Parameter Length:  2
  LOC:  16
  Multi Line of Comment:  0
  Single Line of Comment:  0
```

```
  Method Name:  get_perturbed_actor_updates
    Parameter Length:  2
    LOC:  16
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  91
Multi Line of Comment:  0
Single Line of Comment:  1

Class Name:  DDPG
  Method Number:  19
  Method Name:  setup_target_network_updates
    Parameter Length:  0
    LOC:  6
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  setup_param_noise
    Parameter Length:  1
    LOC:  17
    Multi Line of Comment:  0
    Single Line of Comment:  2
    Comment Percentage:  0.16666666666666666
  Method Name:  setup_actor_optimizer
    Parameter Length:  0
    LOC:  11
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  setup_critic_optimizer
    Parameter Length:  0
    LOC:  22
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  setup_popart
    Parameter Length:  0
    LOC:  18
    Multi Line of Comment:  0
    Single Line of Comment:  1
    Comment Percentage:  0.06666666666666667
  Method Name:  setup_stats
    Parameter Length:  0
    LOC:  36
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  step
    Parameter Length:  3
    LOC:  21
    Multi Line of Comment:  0
    Single Line of Comment:  0
```

```
  Method Name:  _run
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  test_popart
    Parameter Length:  -1
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  test_noise_normal
    Parameter Length:  -1
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  test_noise_ou
    Parameter Length:  -1
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  test_noise_adaptive
    Parameter Length:  -1
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  265
Multi Line of Comment:  0
Single Line of Comment:  18

Class Name:  AdaptiveParamNoiseSpec
  Method Number:  4
  Method Name:  __init__
    Parameter Length:  3
    LOC:  7
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  adapt
    Parameter Length:  1
    LOC:  8
    Multi Line of Comment:  0
    Single Line of Comment:  2
    Comment Percentage:  0.4
  Method Name:  get_stats
    Parameter Length:  0
    LOC:  6
    Multi Line of Comment:  0
    Single Line of Comment:  0
```

```
  Method Name:  __repr__
    Parameter Length:  0
    LOC:  5
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  26
Multi Line of Comment:  0
Single Line of Comment:  2

Class Name:  ActionNoise
  Method Number:  1
  Method Name:  reset
    Parameter Length:  0
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  4
Multi Line of Comment:  0
Single Line of Comment:  0

Class Name:  NormalActionNoise
  Method Number:  3
  Method Name:  __init__
    Parameter Length:  2
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __call__
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __repr__
    Parameter Length:  0
    LOC:  4
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  11
Multi Line of Comment:  0
Single Line of Comment:  0

Class Name:  OrnsteinUhlenbeckActionNoise
  Method Number:  5
  Method Name:  __init__
    Parameter Length:  5
    LOC:  8
    Multi Line of Comment:  0
    Single Line of Comment:  0
```

```
Class Name:  OrnsteinUhlenbeckActionNoise
  Method Number:  5
  Method Name:  __init__
    Parameter Length:  5
    LOC:  8
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __call__
    Parameter Length:  0
    LOC:  5
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  reset
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  __repr__
    Parameter Length:  0
    LOC:  3
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
  Method Name:  as_scalar
    Parameter Length:  0
    LOC:  9
    Multi Line of Comment:  0
    Single Line of Comment:  0
    Comment Percentage:  0.0
LOC:  28
Multi Line of Comment:  0
Single Line of Comment:  0
```

```
Model
Actor
Critic
RingBuffer
Memory
DDPG
AdaptiveParamNoiseSpec
ActionNoise
NormalActionNoise
OrnsteinUhlenbeckActionNoise
{'Actor', 'Model', 'ActionNoise', 'Memory', 'AdaptiveParamNoiseSpec', 'OrnsteinUhlenbeckActionNoise', 'Critic', 'RingBuffer', 'DDPG', 'NormalActionNoise'}
{'Actor', 'Model', 'ActionNoise', 'Memory', 'AdaptiveParamNoiseSpec', 'OrnsteinUhlenbeckActionNoise', 'Critic', 'RingBuffer', 'DDPG', 'NormalActionNoise'}
                        Model  Actor  Critic  RingBuffer  Memory  DDPG  AdaptiveParamNoiseSpec  ActionNoise  NormalActionNoise  OrnsteinUhlenbeckActionNoise
Model                       0      0       0           0       0     0                       0            0                  0                             0
Actor                       0      0       0           0       0     0                       0            0                  0                             0
Critic                      0      0       0           0       0     0                       0            0                  0                             0
RingBuffer                  0      0       0           0       0     0                       0            0                  0                             0
Memory                      0      0       0           0       0     0                       0            0                  0                             0
DDPG                        0      0       0           0       0     0                       0            0                  0                             0
AdaptiveParamNoiseSpec      0      0       0           0       0     0                       0            0                  0                             0
ActionNoise                 0      0       0           0       0     0                       0            0                  0                             0
NormalActionNoise           0      0       0           0       0     0                       0            0                  0                             0
OrnsteinUhlenbeckActionNoise      0      0       0           0       0     0                       0            0                  0                             0
Found
Found
Found
Found
Found
```