Report

제출일	2019년 6월 14일
과목명	오픈소스SW활용
담당교수	황 두 성 교 수 님
전공	공대 소프트웨어학과
학번	32144548
이름	조 창 연

Problem 1

Implement a VotingMachine class that can be used for a simple election. Have methods to clear the machine state, to vote for a Democrat, to vote for a Republican, and to get the tallies for both parties.

<Source Code>

```python
class VotingMachine(object):
    def __init__(self):
        self.democratVotes = 0
        self.republicanVotes = 0

    def voteClear(self):
        self.democratVotes = 0
        self.republicanVotes = 0
        print('clear')

    def voteDemocrat(self):
        self.democratVotes += 1

    def voteRepublican(self):
        self.republicanVotes += 1

    def getVote(self):
        print('Democrat Votes: ', self.democratVotes)
        print('Republican Votes: ', self.republicanVotes)

if __name__ == '__main__':
    vm = VotingMachine()
    vm.getVote()

    vm.voteDemocrat()
    vm.voteDemocrat()
    vm.voteDemocrat()
    vm.voteDemocrat()

    vm.voteRepublican()
    vm.voteRepublican()
    vm.voteRepublican()

    vm.getVote()
    vm.voteClear()
    vm.getVote()
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw/HW2$ python3 p1.py
Democrat Votes:  0
Republican Votes:  0
Democrat Votes:  4
Republican Votes:  3
clear
Democrat Votes:  0
Republican Votes:  0
```

Problem 2

Provide a class Letter for authoring a simple letter. In the constructor, supply the names of the sender and the recipient:
def __init__(self, letterfrom, letterto)

Supply a method
def addLine(self, line)

to add a line of text to the body of the letter. Supply a method
def get_text(self)

that returns the entire text of the letter. The text has the form:
Dear recipient name:

first line of the body

second line of the body

...

last line of the body

Sincerely,

sender name

Test the class with a driver program that prints the following letter.
Dear Dankook:

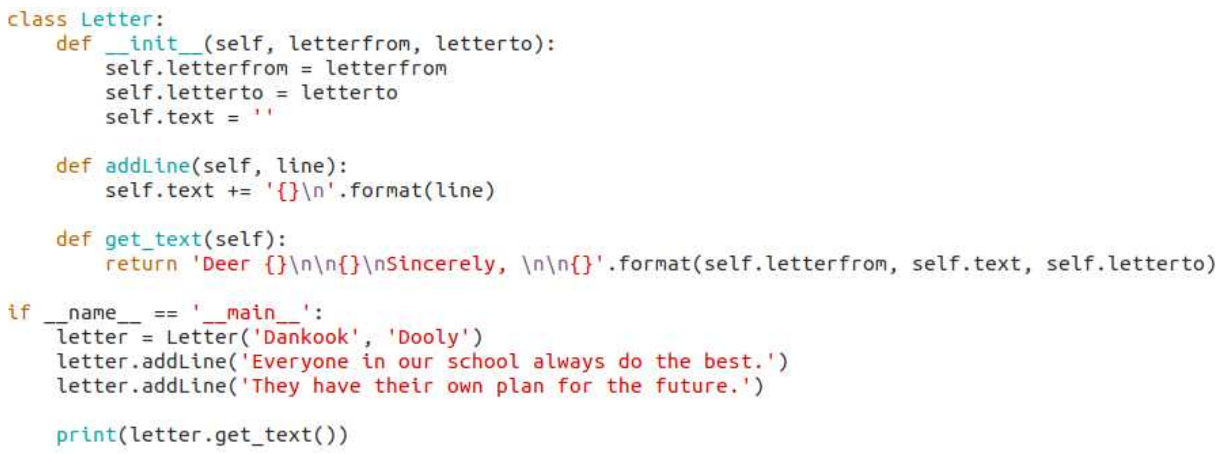
Everyone in our school always do the best.

They have their own plan for the future.

Sincerely,

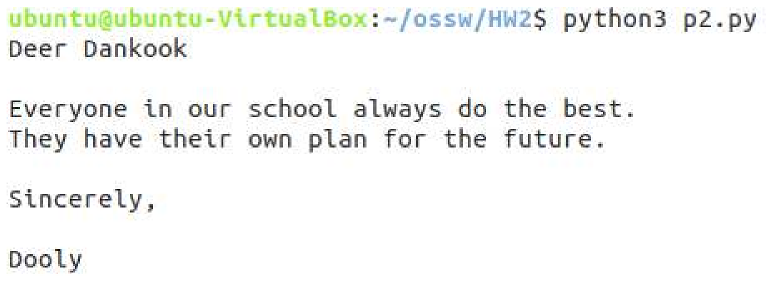
Dooly

<Source Code>



```python
class Letter:
    def __init__(self, letterfrom, letterto):
        self.letterfrom = letterfrom
        self.letterto = letterto
        self.text = ''

    def addLine(self, line):
        self.text += '{}\n'.format(line)

    def get_text(self):
        return 'Deer {}\n\n{}\nSincerely, \n\n{}'.format(self.letterfrom, self.text, self.letterto)

if __name__ == '__main__':
    letter = Letter('Dankook', 'Dooly')
    letter.addLine('Everyone in our school always do the best.')
    letter.addLine('They have their own plan for the future.')

    print(letter.get_text())
```

<Result>



```
ubuntu@ubuntu-VirtualBox:~/ossw/HW2$ python3 p2.py
Deer Dankook

Everyone in our school always do the best.
They have their own plan for the future.

Sincerely,

Dooly
```

Problem 3

Write functions

def sphere_volume(r)

def sphere_surface(r)

def cylinder_volume(r, h)

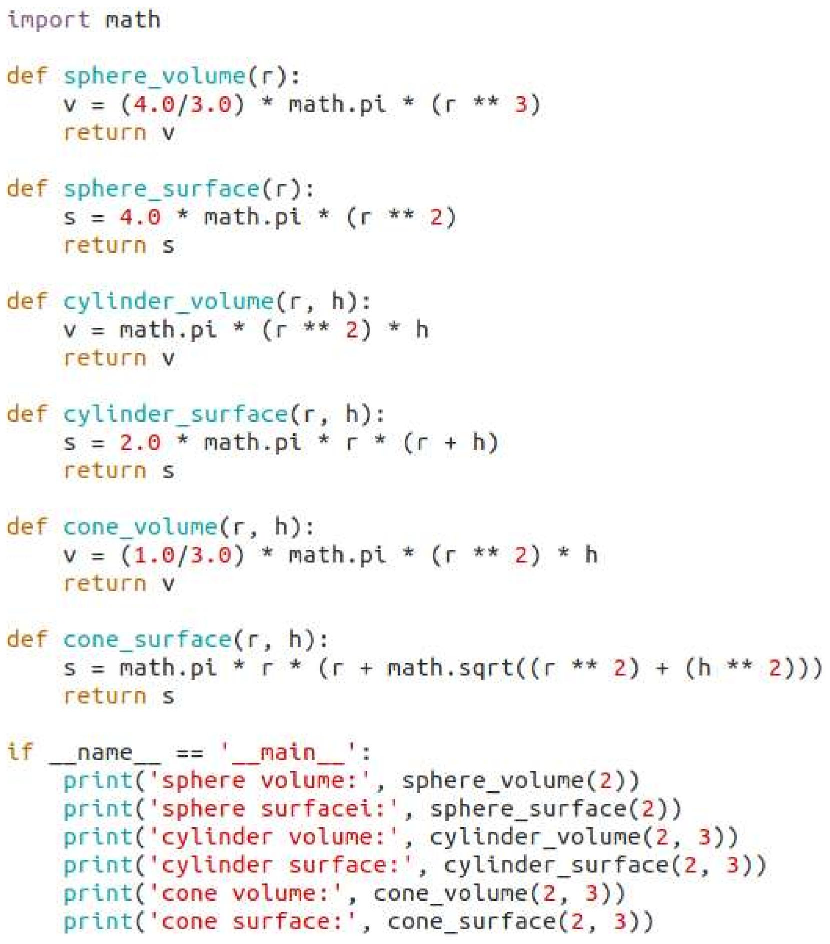
def cylinder_surface(r, h)

def cone_volume(r, h)

def cone_surface(r, h)

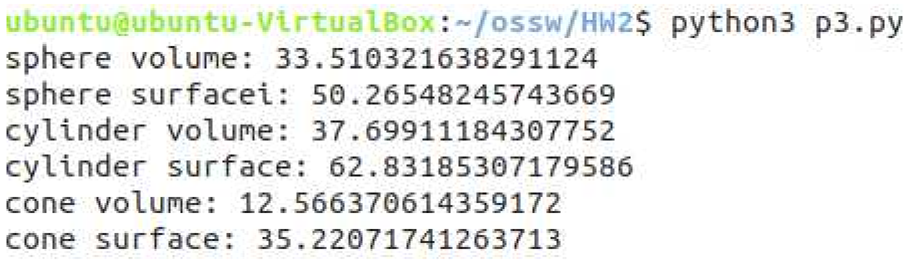
that compute the volume and surface area of a sphere with radius r, a cylinder with a circular base with radius r and height h, and a cone with a circular base with radius r and height h. Place them into a geometry module. Then write a driver program that prompts the user for the values of r and h, calls the six functions, and prints the results.

<Source Code>



```python
import math

def sphere_volume(r):
    v = (4.0/3.0) * math.pi * (r ** 3)
    return v

def sphere_surface(r):
    s = 4.0 * math.pi * (r ** 2)
    return s

def cylinder_volume(r, h):
    v = math.pi * (r ** 2) * h
    return v

def cylinder_surface(r, h):
    s = 2.0 * math.pi * r * (r + h)
    return s

def cone_volume(r, h):
    v = (1.0/3.0) * math.pi * (r ** 2) * h
    return v

def cone_surface(r, h):
    s = math.pi * r * (r + math.sqrt((r ** 2) + (h ** 2)))
    return s

if __name__ == '__main__':
    print('sphere volume:', sphere_volume(2))
    print('sphere surfacei:', sphere_surface(2))
    print('cylinder volume:', cylinder_volume(2, 3))
    print('cylinder surface:', cylinder_surface(2, 3))
    print('cone volume:', cone_volume(2, 3))
    print('cone surface:', cone_surface(2, 3))
```

<Result>

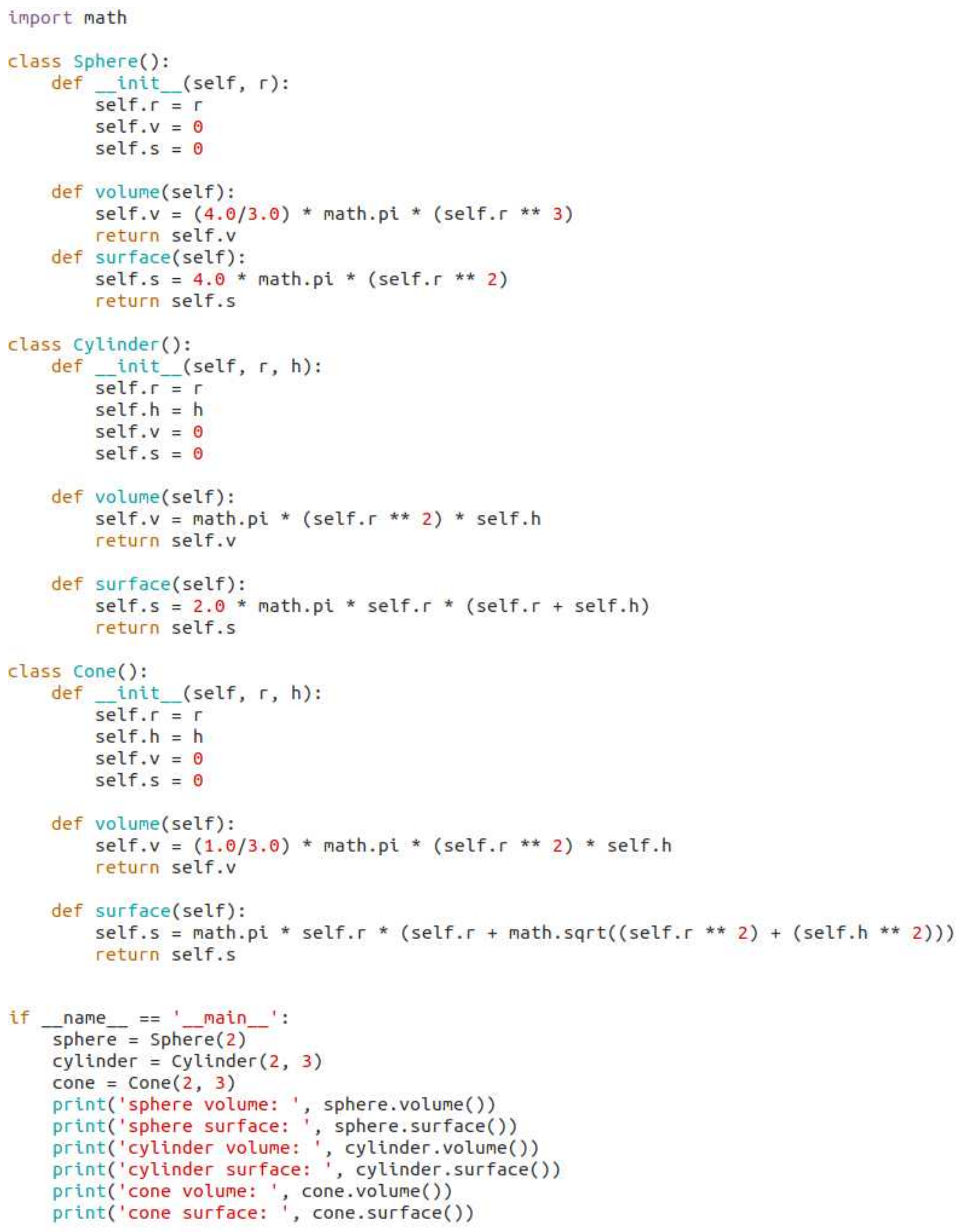


```
ubuntu@ubuntu-VirtualBox:~/ossw/HW2$ python3 p3.py
sphere volume: 33.510321638291124
sphere surfacei: 50.26548245743669
cylinder volume: 37.69911184307752
cylinder surface: 62.83185307179586
cone volume: 12.566370614359172
cone surface: 35.22071741263713
```

Problem 4

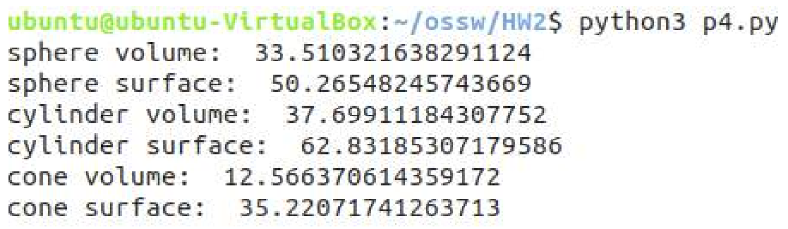
Solve Problem 3 by implementing classes Sphere, Cylinder, and Cone. Write a driver program that tests these classes.

<Source Code>



```python
import math

class Sphere():
    def __init__(self, r):
        self.r = r
        self.v = 0
        self.s = 0

    def volume(self):
        self.v = (4.0/3.0) * math.pi * (self.r ** 3)
        return self.v
    def surface(self):
        self.s = 4.0 * math.pi * (self.r ** 2)
        return self.s

class Cylinder():
    def __init__(self, r, h):
        self.r = r
        self.h = h
        self.v = 0
        self.s = 0

    def volume(self):
        self.v = math.pi * (self.r ** 2) * self.h
        return self.v

    def surface(self):
        self.s = 2.0 * math.pi * self.r * (self.r + self.h)
        return self.s

class Cone():
    def __init__(self, r, h):
        self.r = r
        self.h = h
        self.v = 0
        self.s = 0

    def volume(self):
        self.v = (1.0/3.0) * math.pi * (self.r ** 2) * self.h
        return self.v

    def surface(self):
        self.s = math.pi * self.r * (self.r + math.sqrt((self.r ** 2) + (self.h ** 2)))
        return self.s


if __name__ == '__main__':
    sphere = Sphere(2)
    cylinder = Cylinder(2, 3)
    cone = Cone(2, 3)
    print('sphere volume: ', sphere.volume())
    print('sphere surface: ', sphere.surface())
    print('cylinder volume: ', cylinder.volume())
    print('cylinder surface: ', cylinder.surface())
    print('cone volume: ', cone.volume())
    print('cone surface: ', cone.surface())
```

<Result>



```
ubuntu@ubuntu-VirtualBox:~/ossw/HW2$ python3 p4.py
sphere volume:  33.510321638291124
sphere surface:  50.26548245743669
cylinder volume:  37.69911184307752
cylinder surface:  62.83185307179586
cone volume:  12.566370614359172
cone surface:  35.22071741263713
```

Problem 4

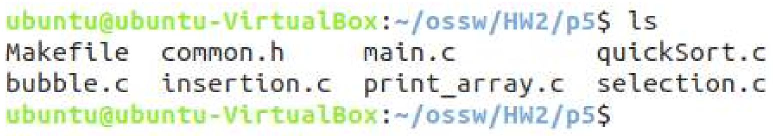
1. Does your code give the same result when you input the same values?
   동일한 결과가 나옵니다.
2. Discuss which approach is more object-oriented.

Problem 3와 비교하면 확실히 Problem 4가 더 object-oriented하다. Problem 3은 일반적인 함수를 호출하여 main으로 해당 함수를 불러오는 개념이라면, Problem 4는 각 항목별로 객체라는 것을 구현하기 위해 class라는 키워드를 사용하였으며, 각 class마다 __init__함수를 사용해 생성자를 만든 후, 함수를 호출하여 main으로 해당 class를 불러오는 개념이다. 큰 차이라고 한다면 Problem 3의 소스코드는 일일이 하나하나 체크를 해봐야 하는 단점이 있다면, Problem 4의 소스코드에선 class라는 하나의 객체를 구현하고, 내부에 필요 mathod를 선언하여 main에서 호출해주기 때문에 Problem 4보단 확실히 깔끔한 코드라고 할 수 있다.

Problem 5

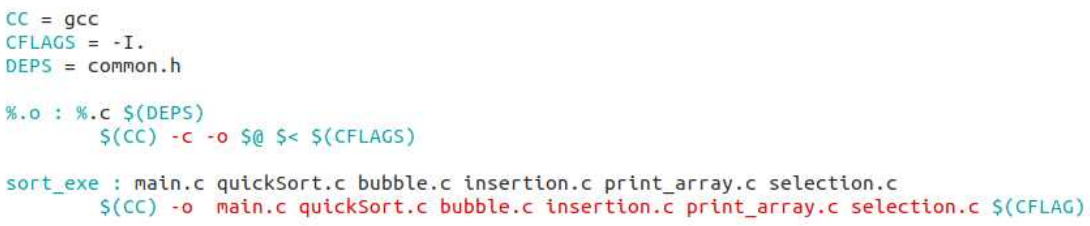
Test the C program(test.c) to sort array. Split the program into more than 4 files and write a Makefile to generate an execution file. Show the test result and check if your result is the same as that of test.c.

<List>



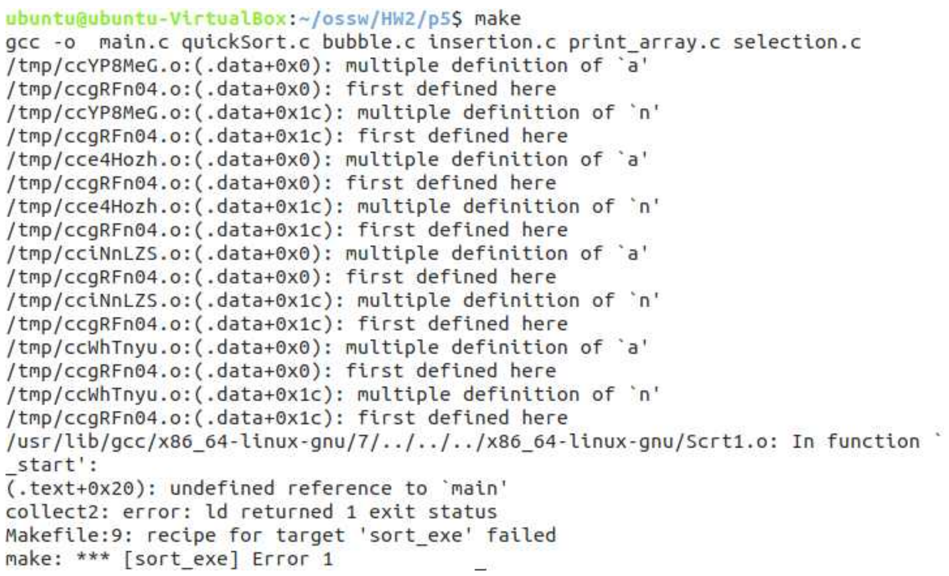
```
ubuntu@ubuntu-VirtualBox:~/ossw/HW2/p5$ ls
Makefile   common.h      main.c         quickSort.c
bubble.c   insertion.c   print_array.c  selection.c
ubuntu@ubuntu-VirtualBox:~/ossw/HW2/p5$
```

<Makefile>



```
CC = gcc
CFLAGS = -I.
DEPS = common.h

%.o : %.c $(DEPS)
        $(CC) -c -o $@ $< $(CFLAGS)

sort_exe : main.c quickSort.c bubble.c insertion.c print_array.c selection.c
        $(CC) -o  main.c quickSort.c bubble.c insertion.c print_array.c selection.c $(CFLAG)
```

<result>



```
ubuntu@ubuntu-VirtualBox:~/ossw/HW2/p5$ make
gcc -o  main.c quickSort.c bubble.c insertion.c print_array.c selection.c
/tmp/ccYP8MeG.o:(.data+0x0): multiple definition of `a'
/tmp/ccgRFn04.o:(.data+0x0): first defined here
/tmp/ccYP8MeG.o:(.data+0x1c): multiple definition of `n'
/tmp/ccgRFn04.o:(.data+0x1c): first defined here
/tmp/cce4Hozh.o:(.data+0x0): multiple definition of `a'
/tmp/ccgRFn04.o:(.data+0x0): first defined here
/tmp/cce4Hozh.o:(.data+0x1c): multiple definition of `n'
/tmp/ccgRFn04.o:(.data+0x1c): first defined here
/tmp/cciNnLZS.o:(.data+0x0): multiple definition of `a'
/tmp/ccgRFn04.o:(.data+0x0): first defined here
/tmp/cciNnLZS.o:(.data+0x1c): multiple definition of `n'
/tmp/ccgRFn04.o:(.data+0x1c): first defined here
/tmp/ccWhTnyu.o:(.data+0x0): multiple definition of `a'
/tmp/ccgRFn04.o:(.data+0x0): first defined here
/tmp/ccWhTnyu.o:(.data+0x1c): multiple definition of `n'
/tmp/ccgRFn04.o:(.data+0x1c): first defined here
/usr/lib/gcc/x86_64-linux-gnu/7/../../../x86_64-linux-gnu/Scrt1.o: In function `
_start':
(.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
Makefile:9: recipe for target 'sort_exe' failed
make: *** [sort_exe] Error 1
```

Problem 5

ppt 자료를 참고하여 Makefile을 작성하여 실행 파일을 생성하려고 하였으나, 위 그림과 같이 같은 오류가 반복되어가고 있습니다. ppt 자료가 잘못되었는지, 아니면 제공해주신 소스코드에 오류가 있었는지 혹은 제 컴퓨터 설정이 이상했었는지 등등 정확한 오류 원인은 모르겠습니다. 구글을 통해 많은 정보를 찾아보아도 뭐가 잘못되었는지 여전히 파악을 하지 못하고 있어서 6번 문제를 이어서 풀기가 어려울 것 같습니다.

Problem 6

Test the C program(test.c) to sort array. Split the program into more than 4 files and write a CMakeLists.txt to generate an execution file. Show the test result and check if your result is the same as that of test.c.