



Inspiring Excellence

Course Code:	CSE111
Course Title:	Programming Language II
Lab No:	07
Topic:	OOP (Inheritance)
Number of tasks:	8

Task 1

The tea company **Kazi and Kazi (KK)** has decided to produce a new line of flavored teas. Design the **KK_tea (parent)** and **KK_flavoured_tea (child)** classes so that the following output is produced. The **KK_flavoured_tea** class should inherit **KK_tea**. Note that:

- An object of either class represents a **single box of teabags**.
- Each tea bag **weighs 2 grams**.
- The **status** of an object refers to whether it is sold or not

Hint: you should use class methods/variables

<pre>t1 = KK_tea(250) print("-----1-----") t1.product_detail() print("-----2-----") KK_tea.total_sales() print("-----3-----") t2 = KK_tea(470, 100) t3 = KK_tea(360, 75) KK_tea.update_sold_status_regular(t1, t2, t3) print("-----4-----") t3.product_detail() print("-----5-----") KK_tea.total_sales() print("-----6-----") t4 = KK_flavoured_tea("Jasmine", 260, 50) t5 = KK_flavoured_tea("Honey Lemon", 270, 45) t6 = KK_flavoured_tea("Honey Lemon", 270, 45) print("-----7-----") t4.product_detail() print("-----8-----") t6.product_detail() print("-----9-----") KK_flavoured_tea.update_sold_status_flavoured(t4, t5, t6) print("-----10-----") KK_tea.total_sales()</pre>	<pre>-----1----- Name: KK Regular Tea, Weight: 100 Tea Bags: 50, Price: 250 Status: False -----2----- Total sales: {'KK Regular Tea': 0} -----3----- -----4----- Name: KK Regular Tea, Weight: 150 Tea Bags: 75, Price: 360 Status: True -----5----- Total sales: {'KK Regular Tea': 3} -----6----- -----7----- Name: KK Jasmine Tea, Weight: 100 Tea Bags: 50, Price: 260 Status: False -----8----- Name: KK Honey Lemon Tea, Weight: 90 Tea Bags: 45, Price: 270 Status: False -----9----- -----10----- Total sales: {'KK Regular Tea': 3, 'KK Jasmine Tea': 1, 'KK Honey Lemon Tea': 2}</pre>
--	--

Task 2

Given the following classes, write the code for the **Cricket_Tournament** and the **Tennis_Tournament** class so that the following output is printed.

```
class Tournament:
    def __init__(self,name='Default'):
        self.__name = name
    def set_name(self,name):
        self.__name = name
    def get_name(self):
        return self.__name

#write your code here

ct1 = Cricket_Tournament()
print(ct1.detail())
print("-----")
ct2 = Cricket_Tournament("IPL",10,"t20")
print(ct2.detail())
print("-----")
tt = Tennis_Tournament("Roland Garros",128)
print(tt.detail())
```

```
OUTPUT:
Cricket Tournament Name: Default
Number of Teams: 0
Type: No type
-----
Cricket Tournament Name: IPL
Number of Teams: 10
Type: t20
-----
Tennis Tournament Name: Roland Garros
Number of Players: 128
```

Task 3

Given the following classes, write the code for the **Triangle** and the **Trapezoid** class so that the following output is printed.

```
class Shape:

    def __init__(self, name='Default', height=0, base=0):
        self.area = 0
        self.name = name
        self.height = height
        self.base = base

    def get_height_base(self):
        return "Height: "+str(self.height)+",Base: "+str(self.base)

#write your code here

tri_default = triangle()
tri_default.calcArea()
```

```
OUTPUT:
Shape name: Default
Height: 0, Base: 0
Area: 0.0
-----
Shape name: Triangle
Height: 10, Base: 5
Area: 25.0
-----
Shape name: Trapezoid
Height: 10, Base: 6, Side_A: 4
Area: 50.0
```

```

print(tri_default.printDetail())
print('-----')
tri = triangle('Triangle', 10, 5)
tri.calcArea()
print(tri.printDetail())
print('-----')
trap = trapezoid('Trapezoid', 10, 6, 4)
trap.calcArea()
print(trap.printDetail())

```

Task 4

Write the **Mango** and the **Jackfruit** classes so that the following code generates the output below:

```

class Fruit:
    def __init__(self, formalin=False, name=''):
        self.__formalin = formalin
        self.name = name

    def getName(self):
        return self.name

    def hasFormalin(self):
        return self.__formalin

class testFruit:
    def test(self, f):
        print('----Printing Detail----')
        if f.hasFormalin():
            print('Do not eat the',f.getName(),'.')
            print(f)
        else:
            print('Eat the',f.getName(),'.')
            print(f)

m = Mango()
j = Jackfruit()
t1 = testFruit()
t1.test(m)
t1.test(j)

```

OUTPUT:

```

----Printing Detail----
Do not eat the Mango.
Mangos are bad for you
----Printing Detail----
Eat the Jackfruit.
Jackfruits are good for you

```

Task 5

A multinational company has two special types of regular employees. One is Foreign employees and another one is Part time employees. Design the Employee (parent), Foreign_employee(child) and Parttime_employee(child) classes so that the following output is produced. The Foreign_employee and Parttime_employee classes should inherit the Employee class. Note that:

- Basic salary of a Regular, Foreign employee is 30,000 and for Part-time employees basic is 15,000.
- Regular employees get 10% increment on their salary and Foreign employees get 15% increment on their basic salary.
- Employees from the HR department will collect their work distribution load from the manager, and others will collect their work distribution load from the HR department.

Driver Code	Output
<pre> print("1-----") emp1=Employee("Nawaz Ali", 102, "Marketing") print("2-----") emp1.employeeDetails() print("3-----") emp1.workDistribution("Marketing") print("4-----") emp1.increment() emp1.employeeDetails() print("5-----") f_emp=Foreign_employee("Nadvi", 311, "Human Resource") f_emp.employeeDetails() print("6-----") f_emp.workDistribution("Human Resource") print("7-----") f_emp.increment() f_emp.employeeDetails() print("8-----") p1_emp=Part_time_employee("Asif", 210, "Sales") p2_emp=Part_time_employee("Olive", 223, "Accounts") print("9-----") p1_emp.employeeDetails() print("10-----") p1_emp.workDistribution("Sales") print("11-----") p2_emp.increment() print("12-----") p2_emp.employeeDetails()</pre>	<pre> 1----- 2----- Name: Nawaz Ali, Dept Marketing Employee id: 102, Salary: 30000 3----- Collect work distribution loads from the HR department. 4----- Name: Nawaz Ali, Dept Marketing Employee id: 102, Salary: 33000.0 5----- Name: Nadvi, Dept Human Resource Employee id: 311, Salary: 30000 Employee Type: Foreign 6----- Collect work distribution details from the manager. 7----- Name: Nadvi, Dept Human Resource Employee id: 311, Salary: 34500.0 Employee Type: Foreign 8----- 9----- Name: Asif, Dept Sales Employee id: 210, Salary: 15000 Employee Type: Part Time 10----- Collect work distribution loads from the HR department. 11----- Sadly, there is no increment for the part time employees!! 12-----</pre>

Name: Olive, Dept Accounts Employee id: 223, Salary: 15000 Employee Type: Part Time

Task 6

You are given the parent class Point:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
        self.area = 0

    def calculate_area(self):
        return self.area

    def print_details(self):
        print("----- Printing details -----")
        print(f'Co-ordinate: ({self.x},{self.y})')
        print(f'Area: {self.area}')
```

Some information about calculating the area of circle and sphere:

Area of a circle: πr^2

Area of a sphere: $4\pi r^2$

Here, Inheritance tree will be Point=>Circle=>Sphere

Write **Circle** and **Sphere** classes to generate the following output.

Driver Code	Output
<pre>print("-----1-----") p1 = Point(2,3) print(f'Area of p1: {p1.calculate_area()}') print("-----2-----") p1.print_details() print("-----3-----")</pre>	<pre>-----1----- Area of p1: 0 -----2----- ----- Printing details ----- Co-ordinate: (2,3) Area: 0 -----3----- ----- Printing details ----- Co-ordinate: (0,0)</pre>

<pre> p2 = Point() p2.print_details() print("-----4-----") c1 = Circle(4,0,3) print(f'Area of c1: {c1.calculate_area()}') print("-----5-----") c1.print_details() print("-----6-----") c2 = Circle(7) print(f'Area of c2: {c2.calculate_area()}') print("-----7-----") sph1 = Sphere(3,0,2) print(f'Area of sph1: {sph1.calculate_area()}') print("-----8-----") sph1.print_details() print("-----9-----") sph2 = Sphere(6) print(f'Area of sph2: {sph2.calculate_area()}') </pre>	<pre> Area: 0 -----4----- Area of c1: 50.2656 -----5----- ----- Printing details ----- Co-ordinate: (0,3) Area: 50.2656 Radius: 4 -----6----- Area of c2: 153.9384 -----7----- Area of sph1: 113.0976 -----8----- ----- Printing details ----- Co-ordinate: (0,2) Area: 113.0976 Radius: 3 -----9----- Area of sph2: 452.3904 </pre>
--	--

Task 7

A bank has two types of accounts : Savings account and Fixed-deposit account. Some features of these accounts are:

- Savings account:
 - An interest rate can be applied
 - You can deposit money anytime you want.
 - Withdrawal can be made unless its crosses the lower limit of the account
- Fixed deposits account:
 - You can not deposit money anytime you want.
 - Withdrawal can be made after the account is matured.

The parent class Account is given below:

class Account:

```

def __init__(self, account_number, balance):
    self.account_number = account_number
    self.balance = balance

```

```

self.account_type = "General"
self.maturity = 0

def print_details(self):
    print("----- Account details -----")
    print(f"Account Type: {self.account_type}, Maturity: {self.maturity} years")
    print(f"Account Number: {self.account_number}, Balance: ${self.balance:.2f}")

def deposit(self, amount):
    self.balance += amount
    print(f"Deposited ${amount:.2f}. New Balance: ${self.balance:.2f}")

def withdraw(self, amount):
    if self.balance >= amount:
        self.balance -= amount
        print(f"Withdrew ${amount:.2f}. New Balance: ${self.balance:.2f}")
    else:
        print("Insufficient funds.")

def year_passed(self, year):
    self.maturity += year
    print(f"Maturity of the account: {self.maturity} years")

```

Write the classes **SavingsAccount** and **FixedDepositAccount** derived from the **Account** class to generate the following output.

Driver Code	Output
<pre> print("-----1-----") account = Account("A203", 2000) account.print_details() print("-----2-----") account.deposit(400) account.withdraw(1500) account.year_passed(2) print("-----3-----") account.print_details() </pre>	<pre> -----1----- ----- Account details ----- Account Type: General, Maturity: 0 years Account Number: A203, Balance: \$2000.00 -----2----- Deposited \$400.00. New Balance: \$2400.00 Withdrew \$1500.00. New Balance: \$900.00 Maturity of the account: 2 years -----3----- ----- Account details ----- Account Type: General, Maturity: 2 years Account Number: A203, Balance: \$900.00 </pre>


```

print("-----4-----")
savings_account = SavingsAccount("Savings","SA123",
1000, 0.05, 500)
savings_account.print_details()
print("-----5-----")
savings_account.deposit(400)
print("-----6-----")
savings_account.withdraw(1000)
print("-----7-----")
savings_account.withdraw(800)
print("-----8-----")
savings_account.apply_interest()
print("-----9-----")
savings_account.print_details()
print("-----10-----")
fixed_account1= FixedDepositAccount("Fixed
Deposit","FDA321", 10000, 5)
fixed_account1.print_details()
print("-----11-----")
fixed_account1.deposit(400)
print("-----12-----")
fixed_account1.year_passed(6)
print("-----13-----")
fixed_account1.withdraw(10000)
print("-----14-----")
fixed_account1.print_details()
print("-----15-----")
fixed_account2 = FixedDepositAccount("Fixed
Deposit","FDA300", 50000, 7)
fixed_account2.print_details()
print("-----16-----")
fixed_account2.withdraw(10000)

```

```

-----4-----
----- Account details -----
Account Type: Savings, Maturity: 0 years
Account Number: SA123, Balance: $1000.00
Interest Rate: 0.05, Minimum Limit: $500
-----5-----
Deposited $400.00. New Balance: $1400.00
-----6-----
Insufficient funds.
-----7-----
Withdrew $800.00. New Balance: $600.00
-----8-----
Interest applied. New Balance: $630.00
-----9-----
----- Account details -----
Account Type: Savings, Maturity: 0 years
Account Number: SA123, Balance: $630.00
Interest Rate: 0.05, Minimum Limit: $500
-----10-----
----- Account details -----
Account Type: Fixed Deposit, Maturity: 0 years
Account Number: FDA321, Balance: $10000.00
-----11-----
You can not deposit in a fixed deposit account.
-----12-----
Maturity of the account: 6 years
-----13-----
Withdrew $10000.00. New Balance: $0.00
-----14-----
----- Account details -----
Account Type: Fixed Deposit, Maturity: 6 years
Account Number: FDA321, Balance: $0.00
-----15-----
----- Account details -----
Account Type: Fixed Deposit, Maturity: 0 years
Account Number: FDA300, Balance: $50000.00
-----16-----
Can not withdraw, Account is not matured

```

Task 8

1	<code>class A:</code>
2	<code> temp = 4</code>
3	<code> def __init__(self):</code>
4	<code> self.sum = 0</code>
5	<code> self.y = 0</code>
6	<code> self.y = A.temp - 2</code>
7	<code> self.sum = A.temp + 1</code>
8	<code> A.temp -= 2</code>
9	<code> def methodA(self, m, n):</code>
10	<code> x = 0</code>
11	<code> self.y = self.y + m + (A.temp)</code>
12	<code> A.temp += 1</code>
13	<code> x = x + 1 + n</code>
14	<code> self.sum = self.sum + x + self.y</code>
15	<code> print(x, self.y, self.sum)</code>
16	
17	<code>class B(A):</code>
18	<code> x = 0</code>
19	<code> def __init__(self,b=None):</code>
20	<code> super().__init__()</code>
21	<code> self.sum = 0</code>
22	<code> if b==None:</code>
23	<code> self.y = A.temp + 3</code>

24	<code>self.sum = 3 + A.temp + 2</code>
25	<code>A.temp -= 2</code>
26	<code>else:</code>
27	<code>self.sum = b.sum</code>
28	<code>B.x = b.x</code>
29	<code>b.methodB(2, 3)</code>
30	<code>def methodB(self, m, n):</code>
31	<code>y = 0</code>
32	<code>y = y + self.y</code>
33	<code>B.x = self.y + 2 + A.temp</code>
34	<code>self.methodA(B.x, y)</code>
35	<code>self.sum = B.x + y + self.sum</code>
36	<code>print(B.x, y, self.sum)</code>

Write the output of the following code:

<pre> a1 = A() b1 = B() b2 = B(b1) b1.methodA(1, 2) b2.methodB(3, 2) </pre>	Output:		
	x	y	sum