



Inspiring Excellence

Course Code:	CSE111
Course Title:	Programming Language II
Classwork No:	07
Topic:	OOP (inheritance)
Number of tasks:	3

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

```
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()
```

```
-----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}
```

Task 2

Given a 2D vector class, design the 3D vector class that inherits 2D vector. You need to implement the following features:

- Similar to X and Y of 2D vector, Z of 3D vector will be a private variable. Hence, write methods that allow access to private variables.
- Write a method **add3DVectors()** that adds 3D vectors. It **must reuse** the **add2DVectors()** function and be written with the same parameters. The only difference is that, in 3D vectors, the Z components are added as well.
- Write a **multiplyScalar()** method that takes an integer as parameter and multiplies it with all 3 components separately (scalar multiplication). Keep in mind that the X and Y variables are private.
- Write a **calculateLength()** that returns the length of the 3D vector using the following formula:
 - $\sqrt{X^2 + Y^2 + Z^2}$
- Write a **print3DVector()** similar to the **print2DVector()** method.
- 2D vector: $Xi + Yj$
3D vector: $Xi + Yj + Zk$

```
class TwoDVector:
    def __init__(self, x, y):
        self.__x = x
        self.__y = y
    # Setter and Getter Methods for x & y
    def add2DVectors(self, *vectors):
        for i in vectors:
            self.__x += i.__x
            self.__y += i.__y
    def print2DVector(self):
        if self.__y >= 0:
            y = "+" + str(self.__y)
        else:
            y = str(self.__y)
        print(f"{self.__x}i {y}j")
```

```

TwoDV1 = TwoDVector(5, 6)
TwoDV2 = TwoDVector(3, 7)
TwoDV3 = TwoDVector(1, 8)
print("=====")
TwoDV1.add2DVectors(TwoDV2, TwoDV3)
TwoDV1.print2DVector()
print("=====")
ThreeDV1 = ThreeDVector(5, 6, 1)
ThreeDV2 = ThreeDVector(1, 9, -7)
ThreeDV3 = ThreeDVector(8, 2, 4)
print("=====")
ThreeDV1.add3DVectors(ThreeDV2, ThreeDV3)
ThreeDV1.print3DVector()
print("=====")
ThreeDV1.multiplyScalar(3)
ThreeDV1.print3DVector()
print("=====")
print(ThreeDV1.calculateLength())

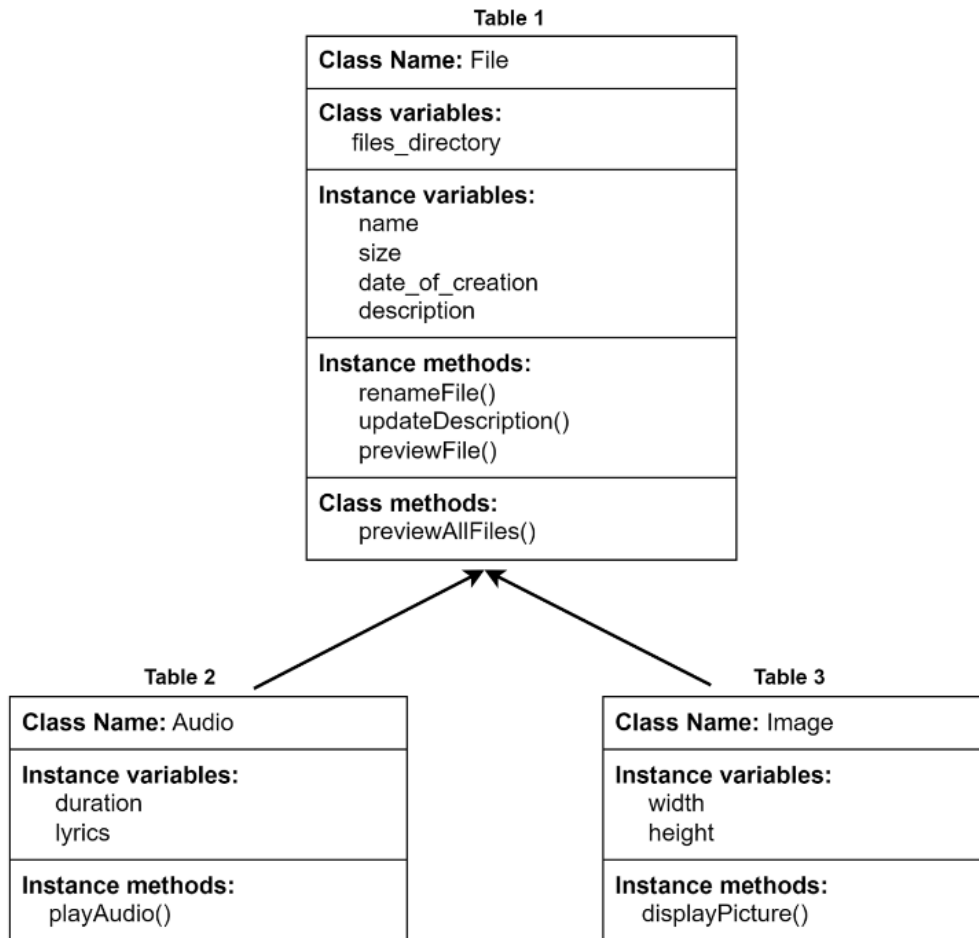
```

```

=====
9i + 21j
=====
=====
14i + 17j -2k
=====
42i + 51j -6k
=====
66.34003316248794

```

Task 3



You are given a **File** class. All the necessary variables and methods are shown in table 1. **previewAllFiles()** is a classmethod which will iterate through each file stored in the class variable **files_directory** and show the file details. You must use the **previewFile()** method to display the file details.

Secondly, implement the **Audio** and **Image** class such that they both **inherit** the File class. All the necessary variables and methods are shown in table 2 and table 3 for Audio and Image classes respectively.

Audio class has a **playAudio()** method **that uses the method implemented in his parent class to show the file details and also shows the lyrics if available.**

In **Image** class, a parameter (width x height) will be given as a string to create an object, but you need to process it to find the **width** (first part of the parameter) and height (second part of the parameter) of the image. You can assume that all images will print a rectangle when the **displayPicture()** method is called.

The driver code for the given scenario and expected output is given [here](#).