|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\Admin\Desktop\download.jpg | USMAN INSTITUTE OF TECHNOLOGY | | | | | |  |
|  |  |  |  |  |  |  |  |
|  | Department of Computer Science  CS121 Object Oriented Programming | | | | | |  |
|  |  | Lab # 08  Class Abstraction and Encapsulation | | | |  |  |
|  | Objective:  This experiment introduces the students to the concept of Class Abstraction and Encapsulation. | | | | | |  |
|  | **Name of Student:**  **Roll No: Sec.**  **Date of Experiment:** | | | | | |  |
|  | **Marks Obtained/Remarks:**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | |  |

**Lab 08: Class Abstraction and Encapsulation**

**Class Abstraction**

Class abstraction is the separation of class implementation from the use of a class. The creator of a class describes the class’s functions and lets the client know how the class can be used. The class’s collection of methods, together with the description of how these methods are expected to behave, serves as the class’s contract with the client.

**Class Encapsulation**

The user of the class does not need to know how the class is implemented. The details of implementation are encapsulated and hidden from the user. This is known as class encapsulation. It is important to note that the class abstraction and encapsulation are two sides of the same coin.

**Real World Example – Building a Computer System**

A personal computer has many components—a CPU, memory, disk, motherboard, fan, and so on. Each component is viewed as an object that has properties and methods. To get the components to work together, we need to know only how each component is used and how it interacts with the others. We don’t need to know how the components work internally.

The internal implementation is encapsulated and hidden from us. We can even build a computer without knowing how a component is implemented.

# Student Exercise

Body Mass Index (BMI)

Body Mass Index (BMI) is a measure of health based on weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters

UML diagram for the BMI class is shown below

|  |
| --- |
| **BMI** |
| - name: String  - age: int  - weight: float  - height: float |
| BMI(name: String, age: int, weight: float, height: float)  getName(): String  getBMI(): float  getStatus(): String |

|  |
| --- |
|  |
| *The name of the person*  *The age of the person*  *The weight of the person in pounds*  *The height of the person in inches* |
| *Constructs a BMI object with the specified name, age, weight, and height.*  *Returns the name*  *Returns the BMI*  *Returns the BMI status (e.g., Normal, Overweight etc.* |

Exercise 1

Write Python code for defining the BMI class

**Code:**

class BMI:

    def \_\_init\_\_(self, name, age, weight, height):

        self.\_\_name = name

        self.\_\_age = age

        self.\_\_weight = weight

        self.\_\_height = height

    def getBMI(self):

        return self.\_\_weight / (self.\_\_height / 100) \*\* 2

    def getStatus(self):

        if self.getBMI() < 18.5:

            return "Underweight"

        elif self.getBMI() < 25:

            return "Normal"

        elif self.getBMI() < 30:

            return "Overweight"

        else:

            return "Obese"

    def getName(self):

        return self.\_\_name

    def getAge(self):

        return self.\_\_age

    def getWeight(self):

        return self.\_\_weight

    def getHeight(self):

        return self.\_\_height

    def setName(self, name):

        self.\_\_name = name

    def setAge(self, age):

        self.\_\_age = age

    def setWeight(self, weight):

        self.\_\_weight = weight

    def setHeight(self, height):

        self.\_\_height = height

    def \_\_str\_\_(self):

        return "Name: {}, Age: {}, Weight: {}, Height: {}, BMI: {}, Status: {}".format(self.\_\_name, self.\_\_age, self.\_\_weight, self.\_\_height, self.getBMI(), self.getStatus())

Exercise 2

Implement a program to create two BMI objects. First one for John Doe, aged 18 years, weighing 145 pounds and having a height of 70 inches. Other for Peter King, aged 50 years, weighing 215 pounds, and having a height of 70 inches. The program then shows the BMI value with status for both the objects along with the name.

**Code:**

# Ex2

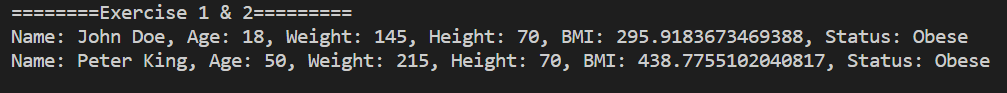
obj1 = BMI("John Doe", 18, 145, 70)

obj2 = BMI("Peter King", 50, 215, 70)

print(obj1)

print(obj2)

**Output:**

****

Geometry: n-sided regular polygon

An n-sided regular polygon’s sides all have the same length and all of its angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named RegularPolygon that contains:

■ A private int data field named n that defines the number of sides in the polygon.

■ A private float data field named side that stores the length of the side.

■ A private float data field named x that defines the x-coordinate of the center of the polygon with default value 0

A private float data field named y that defines the y-coordinate of the center of the polygon with default value 0.

■ A constructor that creates a regular polygon with the specified n (default 3), side (default 1), x (default 0), and y (default 0).

■ The accessor and mutator methods for all data fields.

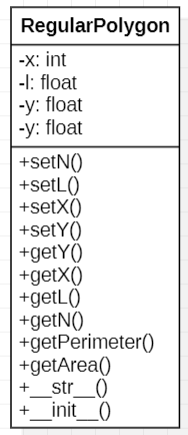
■ The method getPerimeter() that returns the perimeter of the polygon.

■ The method getArea() that returns the area of the polygon. The formula for computing the area of a regular polygon is

Exercise 3

Draw the UML diagram for the class, and then implement the class.

**UML Diagram:**

****

**Code:**

import math

# Exercise 3

class RegularPolygon:

    def \_\_init\_\_(self, n=3, l=1, x=0, y=0):

        self.\_\_n = n

        self.\_\_l = l

        self.\_\_x = x

        self.\_\_y = y

    def setN(self, n):

        self.\_\_n = n

    def setL(self, l):

        self.\_\_l = l

    def setX(self, x):

        self.\_\_x = x

    def setY(self, y):

        self.\_\_y = y

    def getY(self):

        return self.\_\_y

    def getX(self):

        return self.\_\_x

    def getL(self):

        return self.\_\_l

    def getN(self):

        return self.\_\_n

    def getPerimeter(self):

        return self.\_\_n \* self.\_\_l

    def getArea(self):

        return self.\_\_n \* self.\_\_l \*\* 2 / 4 \* math.tan(math.pi / self.\_\_n)

    def \_\_str\_\_(self):

        return "n: {}, l: {}, x: {}, y: {}".format(self.\_\_n, self.\_\_l, self.\_\_x, self.\_\_y)

poly = RegularPolygon()

poly.setN(6)

poly.setL(4)

poly.setX(5.6)

poly.setY(7.8)

print(poly)

print(poly.getPerimeter())

print(poly.getArea())

poly2 = RegularPolygon(10, 4, 5.6, 7.8)

print(poly2)

print(poly2.getPerimeter())

print(poly2.getArea())

**Output:**

Text

Description automatically generated

Exercise 4

Write a test program that creates three RegularPolygon objects, created using RegularPolygon(), using RegularPolygon(6, 4) and RegularPolygon(10, 4, 5.6, 7.8). For each object, display its perimeter and area.

**Code:**

import math

class RegularPolygon:

    def \_\_init\_\_(self, n=0, l=0, x=0, y=0):

        self.n = n

        self.l = l

        self.x = x

        self.y = y

    def perimeter(self):

        return self.n \* self.l \* math.pi

    def area(self):

        return self.n \* self.l \*\* 2 / 4 \* math.tan(math.pi / self.n)

    def \_\_str\_\_(self):

        return "n: {}, l: {}, x: {}, y: {}".format(self.n, self.l, self.x, self.y)

obj1 = RegularPolygon()

obj2 = RegularPolygon(6, 4)

obj3 = RegularPolygon(10, 4, 5.6, 7.8)

print(obj1)

print(obj2)

print(obj3)

**Output:**

Text

Description automatically generated