****

**Course Title: Programming Language II**

**Course Code: CSE 111**

**Lab Assignment no: 6**

**Task 1**

Write a **Student** class to get the desired output as shown below.

1. Create a Student class and a class variable called ID initialized with 0.
2. Create a constructor that takes 4 parameters: name, department, age and cgpa.
3. Write a **showDetails()** method to represent all the details of a Student
4. Write a *class method* **from\_String()** that takes 1 parameter which includes name, department, age and cgpa all four attributes in string.

| ***#Write your code here for subtasks 1-6.***  s1 = Student("Samin", "CSE", 21, 3.91)  s1.showDetails()  print("-----------------------")  s2 = Student("Fahim", "ECE", 21, 3.85)  s2.showDetails()  print("-----------------------")  s3 = Student("Tahura", "EEE", 22, 3.01)  s3.showDetails()  print("-----------------------")  s4 = Student.from\_String("Sumaiya-BBA-23-3.96")  s4.showDetails()  ***# Write the answer of subtask 5 here***  ***# Write the answer of subtask 6 here***  **#You are not allowed to change the code above** | **OUTPUT**  ID: 1  Name: Samin  Department: CSE  Age: 21  CGPA: 3.91  -----------------------  ID: 2  Name: Fahim  Department: ECE  Age: 21  CGPA: 3.85  -----------------------  ID: 3  Name: Tahura  Department: EEE  Age: 22  CGPA: 3.01  -----------------------  ID: 4  Name: Sumaiya  Department: BBA  Age: 23  CGPA: 3.96 |
| --- | --- |

1. Explain the difference between a class variable and an instance variable. Print your answer at the very end of your code.
2. What is the difference between an instance method and class method? Print your answer at the very end

**Task 2**

Write the **Assassin** class so that the given code provides the expected output.

1. Create **Assassin** class
2. Create 1 class variable
3. Create 1 class method titled ‘failureRate()’
4. Create 1 class method titled ‘failurePercentage()’
5. Maximum success\_rate is 100

**[You are not allowed to change the code below]**

| ***# Write your code here***    john\_wick = Assassin('John Wick', 100)  john\_wick.printDetails()  print('================================')  nagisa = Assassin.failureRate("Nagisa", 20)  nagisa.printDetails()  print('================================')  akabane = Assassin.failurePercentage("Akabane", "10%")  akabane.printDetails() | ***Output:***  Name: John Wick  Success rate: 100%  Total number of Assassin: 1  ============================  Name: Nagisa  Success rate: 80%  Total number of Assassin: 2  ============================  Name: Akabane  Success rate: 90%  Total number of Assassin: 3 |
| --- | --- |

**Task 3**

Implement the design of the **Passenger** class so that the following output is produced:

The assumption is Bus base-fare is 450 taka. A passenger can carry upto 20 kg for free. 50 taka will be added if bag weight is between 21 and 50 kg. 100 taka will be added if bag weight is greater than 50 kg.

**[You are not allowed to change the code below]**

| ***# Write your code here***    print(“Total Passenger:”, Passenger.count)  p1 = Passenger(“Jack”)  p1.set\_bag\_weight(90)  p2 = Passenger(“Carol”)  p2.set\_bag\_weight(10)  p3 = Passenger(“Mike”)  p3.set\_bag\_weight(25)  print("=========================")  p1.printDetail()  print("=========================")  p2.printDetail()  print("=========================")  p3.printDetail()  print("=========================")  print(“Total Passenger:”, Passenger.count) | **Output:**  Total Passenger: 0  =========================  Name: Jack  Bus Fare: 550 taka  =========================  Name: Carol  Bus Fare: 450 taka  =========================  Name: Mike  Bus Fare: 500 taka  =========================  Total Passenger: 3 |
| --- | --- |

**Task 4**

Implement the design of the **Travel** class so that the following output is produced:

**[You are not allowed to change the code below]**

| ***# Write your code here***    print(“No. of Traveller =”, Travel.count)  print("=======================")  t1 = Travel("Dhaka","India")  print(t1.display\_travel\_info())  print("=======================")  t2 = Travel("Kuala Lampur","Dhaka")  t2.set\_time(23)  print(t2.display\_travel\_info())  print("=======================")  t3 = Travel("Dhaka","New\_Zealand")  t3.set\_time(15)  t3.set\_destination("Germany")  print(t3.display\_travel\_info())  print("=======================")  t4 = Travel("Dhaka","India")  t4.set\_time(9)  t4.set\_source("Malaysia")  t4.set\_destination("Canada")  print(t4.display\_travel\_info())  print("=======================")  print(“No. of Traveller =”, Travel.count) | ***Output***  No. of Traveller = 0  =======================  Source: Dhaka  Destination:India  Flight Time:1:00  =======================  Source: Kuala Lampur  Destination:Dhaka  Flight Time:23:00  =======================  Source: Dhaka  Destination:Germany  Flight Time:15:00  =======================  Source: Malaysia  Destination:Canada  Flight Time:9:00  =======================  No of Traveller = 4 |
| --- | --- |

**Task 5**

Create an **Employee** Class that will have

* Two instance variable: name and workingPeriod
* A class method named employeeByJoiningYear():
  + To create an Employee object by joining year for calculating the working period
  + It will have two Parameter name and year
* A **static method** experienceCheck() to check if an Employee is experienced or not
  + It will take working period and gender as parameter
  + If an employee’s working period is less than 3, he or she is not experienced

**[You are not allowed to change the code below]**

| ***# Write your code here***  employee1 = Employee('Dororo', 3)  employee2 = Employee.employeeByJoiningYear('Harry', 2016)  print(employee1.workingPeriod)  print(employee2.workingPeriod)  print(employee1.name)  print(employee2.name)  print(Employee.experienceCheck(2, "male"))  print(Employee.experienceCheck(3, "female")) | ***Output***  3  7  Dororo  Harry  He is not experienced  She is experienced |
| --- | --- |

**Task 6**

Implement the design of the **Laptop** class so that the following output is produced

**[You are not allowed to change the code below]**

| ***# Write your code here***  lenovo = Laptop("Lenovo", 5)  dell = Laptop("Dell", 7)  print(lenovo.name, lenovo.count)  print(dell.name, dell.count)  print("Total number of Laptops", Laptop.laptopCount)  Laptop.advantage()  Laptop.resetCount()  print("Total number of Laptops", Laptop.laptopCount) | ***Output***  Lenovo 5  Dell 7  Total number of Laptops 12  Laptops are portable  Total number of Laptops 0 |
| --- | --- |

**Task 7**

Design **Cat** class for the following code to get the output as shown.

You have already solved this problem in assignment 4 using constructor overloading. Now, solve this again but this time DO NOT USE CONSTRUCTOR OVERLOADING.

*Hint: You will have to use classmethods.*

**[You are not allowed to change the code below]**

| ***# Write your code here***    print("Total number of cats:", Cat.Number\_of\_cats)  c1 = Cat.no\_parameter()  c2 = Cat.first\_parameter("Black")  c3 = Cat("Brown", "jumping")  c4 = Cat("Red", "purring")  c5 = Cat.second\_parameter("playing")  print("=======================")  c1.printCat()  c2.printCat()  c3.printCat()  c4.printCat()  c5.printCat()  c1.changeColor("Blue")  c3.changeColor("Purple")  c1.printCat()  c3.printCat()  print("=======================")  print("Total number of cats:", Cat.Number\_of\_cats) | **Output:**  Total number of cats: 0  =======================  White cat is sitting  Black cat is sitting  Brown cat is jumping  Red cat is purring  Grey cat is playing  Blue cat is sitting  Purple cat is jumping  =======================  Total number of cats: 5 |
| --- | --- |

**Task 8**

Write a **Cylinder** class to get the desired output as shown below.

1. You will have to create a Cylinder class.
2. You will have to create 2 class variables.
3. Create a required constructor.
4. Write 2 *class methods:*

* One that takes the height first and then the radius and then swaps
* Onethat takes a string where the radius and height values are separated with a hyphen.

Write 2 *static methods:*

* One that calculates the area of a whole cylinder *(formula: 2πr2 + 2πrh)*
* Another that calculates the volume of a cylinder *(formula: πr2h)*

*\*\*Observe the output values carefully to understand how the radius and height values are changing.*

**[You are not allowed to change the code below]**

| ***# Write your code here***    c1 = Cylinder(0,0)  Cylinder.area(c1.radius,c1.height)  Cylinder.volume(c1.radius,c1.height)  print("===============================")  c2 = Cylinder.swap(8,3)  c2.area(c2.radius,c2.height)  c2.volume(c2.radius,c2.height)  print("===============================")  c3 = Cylinder.changeFormat("7-13")  c3.area(c3.radius,c3.height)  c3.volume(c3.radius,c3.height)  print("===============================")  Cylinder(0.3,5.56).area(Cylinder.radius,Cylinder.height)  print("===============================")  Cylinder(3,5).volume(Cylinder.radius,Cylinder.height) | **Output:**  Default radius=5 and height=18.  Updated: radius=0 and height=0.  Area: 0.0  Volume: 0.0  ===============================  Default radius=0 and height=0.  Updated: radius=3 and height=8.  Area: 207.34511513692635  Volume: 226.1946710584651  ===============================  Default radius=3 and height=8.  Updated: radius=7.0 and height=13.0.  Area: 879.645943005142  Volume: 2001.1945203366981  ===============================  Default radius=7.0 and height=13.0.  Updated: radius=0.3 and height=5.56.  Area: 11.045839770021713  ===============================  Default radius=0.3 and height=5.56.  Updated: radius=3 and height=5.  Volume: 141.3716694115407 |
| --- | --- |

**Task 9**

Write the **Student** class so that the given code provides the expected output.

1. Create **Student** class
2. Create 3 class variable
3. Create 1 class method for object creation
4. Create 1 class method for printing

**[You are not allowed to change the code below]**

| ***# Write your code here***    Student.printDetails()  print('#########################')  mikasa = Student('Mikasa Ackerman', "CSE")  mikasa.individualDetail()  print('------------------------------------------')  Student.printDetails()  print('========================')  harry = Student.createStudent('Harry Potter', "Defence Against Dark Arts", "Hogwarts School")  harry.individualDetail()  print('-------------------------------------------')  Student.printDetails()  print('=========================')  levi = Student.createStudent("Levi Ackerman", "CSE")  levi.individualDetail()  print('--------------------------------------------')  Student.printDetails() | ***Output:***  Total Student(s): 0  BRAC University Student(s): 0  Other Institution Student(s): 0  ################################  Name: Mikasa Ackerman  Department: CSE  Institution: BRAC University  ------------------------------------------------------  Total Student(s): 1  BRAC University Student(s): 1  Other Institution Student(s): 0  ===============================  Name: Harry Potter  Department: Defence Against Dark Arts  Institution: Hogwarts School  ------------------------------------------------------  Total Student(s): 2  BRAC University Student(s): 1  Other Institution Student(s): 1  ===============================  Name: Levi Ackerman  Department: CSE  Institution: BRAC University  ------------------------------------------------------  Total Student(s): 3  BRAC University Student(s): 2  Other Institution Student(s): 1 |
| --- | --- |

**Task 10**

Write the **SultansDine** class so that the given code provides the expected output.

**[You are not allowed to change the code below]**

| ***# Write your code here***  SultansDine.details()  print('########################')  dhanmondi = SultansDine('Dhanmondi')  dhanmondi.sellQuantity(25)  dhanmondi.branchInformation()  print('-----------------------------------------')  SultansDine.details()  print('========================')  baily\_road = SultansDine('Baily Road')  baily\_road.sellQuantity(15)  baily\_road.branchInformation()  print('-----------------------------------------')  SultansDine.details()  print('========================')  gulshan = SultansDine('Gulshan')  gulshan.sellQuantity(9)  gulshan.branchInformation()  print('-----------------------------------------')  SultansDine.details() | ***Output:***  Total Number of branch(s): 0  Total Sell: 0 Taka  #################################  Branch Name: Dhanmondi  Branch Sell: 10000 Taka  -----------------------------------------  Total Number of branch(s): 1  Total Sell: 10000 Taka  Branch Name: Dhanmondi, Branch Sell: 10000 Taka  Branch consists of total sell's: 100.00%  ================================  Branch Name: Baily Road  Branch Sell: 5250 Taka  -----------------------------------------  Total Number of branch(s): 2  Total Sell: 15250 Taka  Branch Name: Dhanmondi, Branch Sell: 10000 Taka  Branch consists of total sell's: 65.57%  Branch Name: Baily Road, Branch Sell: 5250 Taka  Branch consists of total sell's: 34.43%  ================================  Branch Name: Gulshan  Branch Sell: 2700 Taka  -----------------------------------------  Total Number of branch(s): 3  Total Sell: 17950 Taka  Branch Name: Dhanmondi, Branch Sell: 10000 Taka  Branch consists of total sell's: 55.71%  Branch Name: Baily Road, Branch Sell: 5250 Taka  Branch consists of total sell's: 29.25%  Branch Name: Gulshan, Branch Sell: 2700 Taka  Branch consists of total sell's: 15.04% |
| --- | --- |

**Subtaks:**

1. Create **SultansDine** class
2. Create 2 class variable and 1 class list
3. Create 1 class method
4. Calculation of branch sell is given below
   1. If sellQuantity < 10:
      1. Branch\_sell = quantity \* 300
   2. Else if sellQuantity < 20:
      1. Branch\_sell = quantity \* 350
   3. Else
      1. Branch\_sell = quantity \* 400
5. Calculation of branch’s sell percentage = (branch’s sell / total sell) \* 100

**Task 11**

| **1** | **class Puzzle:** |
| --- | --- |
| **2** | **x = 0** |
| **3** | **def methodA(self):** |
| **4** | **Puzzle.x = 5** |
| **5** | **z = Puzzle.x + self.methodB(Puzzle.x)** |
| **6** | **print(Puzzle.x, z)** |
| **7** | **z = self.methodB(z + 2) + Puzzle.x** |
| **8** | **print(Puzzle.x, z)** |
| **9** | **self.methodB(Puzzle.x, z)** |
| **10** | **print(Puzzle.x, z)** |
| **11** | **def methodB(self, \*args):** |
| **12** | **if len(args) == 1:** |
| **13** | **y = args[0]** |
| **14** | **Puzzle.x = y + Puzzle.x** |
| **15** | **print(Puzzle.x, y)** |
| **16** | **return Puzzle.x + 3** |
| **17** | **else:** |
| **18** | **z, x = args** |
| **19** | **z = z + 1** |
| **20** | **x = x + 1** |
| **21** | **print(z, x)** |

| **p = Puzzle()**  **p.methodA()**  **p.methodA()**  **p = Puzzle()**  **p.methodA()**  **p.methodB(7)** | Output-1 | Output-2 |
| --- | --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**Task 12**

| **1** | **class FinalT6A:** |
| --- | --- |
| **2** | **temp = 3** |
| **3** |  |
| **4** | **def \_\_init\_\_(self, x, p):** |
| **5** | **self.sum, self.y = 0, 2** |
| **6** | **FinalT6A.temp += 3** |
| **7** | **self.y = self.temp - p** |
| **8** | **self.sum = self.temp + x** |
| **9** | **print(x, self.y, self.sum)** |
| **10** |  |
| **11** | **def methodA(self):** |
| **12** | **x, y = 0, 0** |
| **13** | **y = y + self.y** |
| **14** | **x = self.y + 2 + self.temp** |
| **15** | **self.sum = x + y + self.methodB(self.temp, y)** |
| **16** | **print(x, y, self.sum)** |
| **17** |  |
| **18** | **def methodB(self, temp, n):** |
| **19** | **x = 0** |
| **20** | **FinalT6A.temp += 1** |
| **21** | **self.y = self.y + (FinalT6A.temp)** |
| **22** | **FinalT6A.temp -= 1** |
| **23** | **x = x + 2 + n** |
| **24** | **self.sum = self.sum + x + self.y** |
| **25** | **print(x, self.y, self.sum)** |
| **26** | **return self.sum** |

| **q1 = FinalT6A(2,1)**  **q1.methodA()**  **q1.methodA()** | **x** | **y** | **sum** |
| --- | --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Task 13**

| **1** | **class A:** |
| --- | --- |
| **2** | **temp = 4** |
| **3** | **def \_\_init\_\_(self):** |
| **4** | **self.y = self.temp - 2** |
| **5** | **self.sum = self.temp + 1** |
| **6** | **A.temp -= 2** |
| **7** | **self.methodA(3, 4)** |
| **8** | **def methodA(self, m, n):** |
| **9** | **x = 0** |
| **10** | **self.y = self.y + m + (self.temp)** |
| **11** | **A.temp += 1** |
| **12** | **x = x + 1 + n** |
| **13** | **self.sum = self.sum + x + self.y** |
| **14** | **print(x, self.y, self.sum)** |
| **15** |  |
| **16** | **class B:** |
| **17** | **x = 0** |
| **18** | **def \_\_init\_\_(self, b = None):** |
| **19** | **self.y, self.temp, self.sum = 5, -5, 2** |
| **20** |  |
| **21** | **if b == None:** |
| **22** | **self.y = self.temp + 3** |
| **23** | **self.sum = 3 + self.temp + 2** |
| **24** | **self.temp -= 2** |
| **25** | **else:** |
| **26** | **self.sum = b.sum** |
| **27** | **B.x = b.x** |
| **28** | **b.methodB(2, 3)** |
| **29** | **def methodA(self, m, n):** |
| **30** | **x = 2** |
| **31** | **self.y = self.y + m + (self.temp)** |
| **32** | **self.temp += 1** |
| **33** | **x = x + 5 + n** |
| **34** | **self.sum = self.sum + x + self.y** |
| **35** | **print(x, self.y, self.sum)** |
| **36** | **def methodB(self, m, n):** |
| **37** | **y = 0** |
| **38** | **y = y + self.y** |
| **39** | **B.x = self.y + 2 + self.temp** |
| **40** | **self.methodA(self.x, y)** |
| **41** | **self.sum = self.x + y + self.sum** |
| **42** | **print(self.x, y, self.sum)** |



**Task 14**

| **1** | **class msgClass:** |
| --- | --- |
| **2** | **def \_\_init\_\_(self):** |
| **3** | **self.content = 0** |
| **4** |  |
| **5** | **class Quiz3:** |
| **6** | **x = 0** |
| **7** | **def \_\_init\_\_(self, k = None):** |
| **8** | **self.sum, self.y = 0, 0** |
| **9** | **if k is None:** |
| **10** | **self.sum = 5** |
| **11** | **Quiz3.x = 2** |
| **12** | **self.y = 2** |
| **13** | **else:** |
| **14** | **self.sum = self.sum + k** |
| **15** | **self.y = 3** |
| **16** | **Quiz3.x += 2** |
| **17** | **def methodA(self):** |
| **18** | **x = 1** |
| **19** | **y = 1** |
| **20** | **msg = [None]** |
| **21** | **myMsg = msgClass()** |
| **22** | **myMsg.content = Quiz3.x** |
| **23** | **msg[0] = myMsg** |
| **24** | **msg[0].content = self.y + myMsg.content** |
| **25** | **self.y = self.y + self.methodB(msg[0])** |
| **26** | **y = self.methodB(msg[0]) + self.y** |
| **27** | **x = y + self.methodB(msg, msg[0])** |
| **28** | **self.sum = x + y + msg[0].content** |
| **29** | **print(x, y, self.sum)** |
| **30** | **def methodB(self, \*args):** |
| **31** | **if len(args) == 2:** |
| **32** | **mg2, mg1 = args** |
| **33** | **x = 2** |
| **34** | **self.y = self.y + mg2[0].content** |
| **35** | **mg2[0].content = self.y + mg1.content** |
| **36** | **x = x + 2 + mg1.content** |
| **37** | **self.sum = self.sum + x + self.y** |
| **38** | **mg1.content = self.sum - mg2[0].content** |
| **39** | **print(Quiz3.x, self.y, self.sum)** |
| **40** | **return self.sum** |
| **41** |  |
| **42** | **elif len(args) == 1:** |
| **43** | **mg1, = args** |
| **44** | **x = 1** |
| **45** | **y = 2** |
| **46** | **y = self.sum + mg1.content** |
| **47** | **self.y = y + mg1.content** |
| **48** | **x = Quiz3.x + 5 + mg1.content** |
| **49** | **self.sum = self.sum + x + y** |
| **50** | **Quiz3.x = mg1.content + x + 3** |
| **51** | **print(x, y, self.sum)** |
| **52** | **return y** |

| **a1 = Quiz3()**  **a2 = Quiz3(5)**  **msg = msgClass()**  **a1.methodA()**  **a2.methodB(msg)** | **x** | **y** | **sum** |
| --- | --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |