Exp 9 — Classes, Constructors & Destructors (Solved)

Name: Abhishek Kadadevarmath
Class: TY C
Roll No: C46

1. Student Class: name and marks. methods: display, change school (class method), calculate grade (static), destructor message.

```
class Student:
    school = "ABC School"
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks
    def display(self):
        print(f"Name: {self.name} | Marks: {self.marks} | School:
{Student.school}")
    @classmethod
    def change_school(cls, new):
        cls.school = new
        print("School changed to", cls.school)
    @staticmethod
    def grade(marks):
        avg = sum(marks)/len(marks)
        if avg>=90: return 'A+'
        if avg>=75: return 'A'
        if avg>=50: return 'B'
        return 'C'
    def __del__(self):
        print(f"Student object for {self.name} is destroyed")

s = Student('Alice',[80,90,85])
s.display()
print('Grade:', Student.grade(s.marks))
Student.change_school('New High')
s.display()
del s
```

Output:

```
Name: Alice | Marks: [80, 90, 85] | School: ABC School
Grade: A
School changed to New High
Name: Alice | Marks: [80, 90, 85] | School: New High
Student object for Alice is destroyed
```

2. Employee Salary: emp_id, name, salary. yearly salary, update company name (class), bonus (static), destructor.

```
class Employee:
    company = "ABC Corp"
    def __init__(self, emp_id, name, salary):
        self.emp_id = emp_id
        self.name = name
        self.salary = salary
    def yearly_salary(self):
        return self.salary * 12
    @classmethod
    def update_company(cls, name):
        cls.company = name
        print('Company updated to', cls.company)
    @staticmethod
    def bonus(salary, percent):
        return salary * percent / 100
    def __del__(self):
        print(f"Employee {self.name} removed")

e = Employee(101,'Bob',4000)
print('Yearly:', e.yearly_salary())
print('Bonus 10%:', Employee.bonus(e.salary,10))
Employee.update_company('XYZ Ltd')
print('Company now:', Employee.company)
del e
```

Output:

```
Yearly: 48000
Bonus 10%: 400.0
Company updated to XYZ Ltd
Company now: XYZ Ltd
Employee Bob removed
```

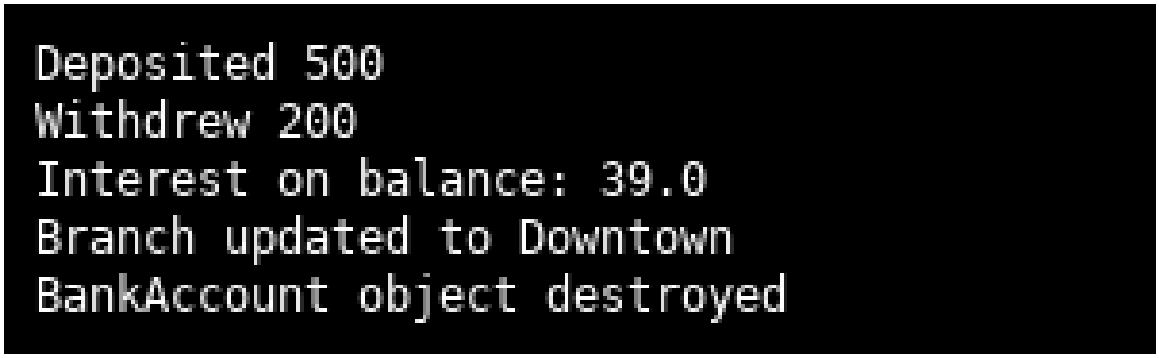
3. Bank Account: acc_no, balance. deposit/withdraw, update branch (class), interest (static), destructor.

```
class BankAccount:
    branch = "Main"
    def __init__(self, acc_no, balance=0):
        self.acc_no = acc_no
        self.balance = balance
    def deposit(self, amt):
        if amt>0:
            self.balance += amt
            print('Deposited', amt)
    def withdraw(self, amt):
        if 0<amt<=self.balance:
            self.balance -= amt
            print('Withdrew', amt)
        else:
            print('Insufficient funds or invalid amount')
    @classmethod
    def update_branch(cls, b):
        cls.branch = b; print('Branch updated to', b)
    @staticmethod
    def interest(balance, rate=0.05):
        return balance * rate
    def __del__(self):
        print('BankAccount object destroyed')

acc = BankAccount('ACC123',1000)
acc.deposit(500)
acc.withdraw(200)
print('Interest on balance:',
BankAccount.interest(acc.balance,0.03))
BankAccount.update_branch('Downtown')
```

```
del acc
```

Output:

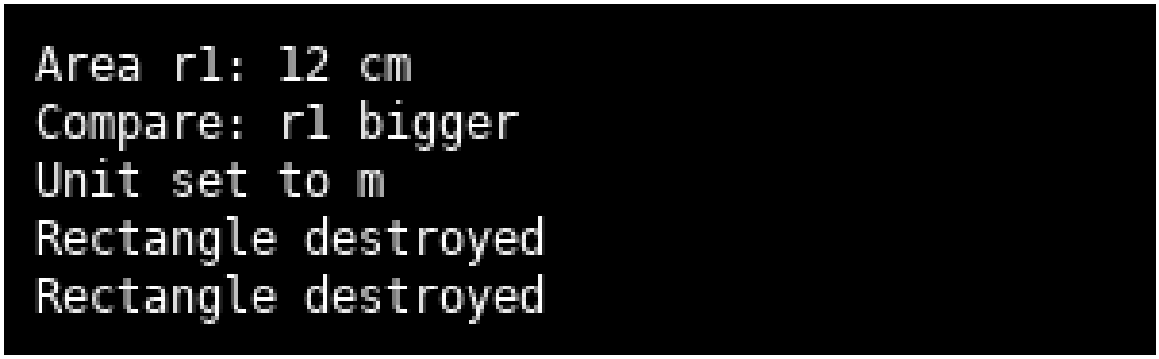


4. Rectangle Operations: length, width. area/perimeter, update unit (class), compare two rectangles (static), destructor.

```
class Rectangle:
    unit = 'cm'
    def __init__(self, l, w):
        self.l = l; self.w = w
    def area(self): return self.l * self.w
    def perimeter(self): return 2*(self.l + self.w)
    @classmethod
    def set_unit(cls, u):
        cls.unit = u; print('Unit set to', u)
    @staticmethod
    def compare(r1, r2):
        if r1.area() > r2.area(): return 'r1 bigger'
        if r1.area() < r2.area(): return 'r2 bigger'
        return 'equal'
    def __del__(self): print('Rectangle destroyed')

r1 = Rectangle(3,4); r2 = Rectangle(5,2)
print('Area r1:', r1.area(), Rectangle.unit)
print('Compare:', Rectangle.compare(r1,r2))
Rectangle.set_unit('m')
del r1; del r2
```

Output:



5. Car Information: brand, model, price. display, update total cars (class), discounted price (static), destructor.

```
class Car:
    total_cars = 0
    def __init__(self, brand, model, price):
        self.brand = brand; self.model = model; self.price =
price
        Car.total_cars += 1
    def display(self): print(f'{self.brand} {self.model} -
{self.price}')
    @classmethod
    def update_total(cls, n): cls.total_cars = n; print('Total
cars set to', n)
    @staticmethod
    def discounted(price, percent): return price * (100-
percent)/100
    def __del__(self): print('Car object destroyed')

c = Car('Honda','City',12000); c.display()
print('Discounted:', Car.discounted(c.price,10))
Car.update_total(50)
del c
```

Output:

```
Honda City - 12000
Discounted: 10800.0
Total cars set to 50
Car object destroyed
```

6. Circle Calculations: radius. area/circumference, update pi (class), validate radius (static), destructor.

```
class Circle:
    PI = math.pi
    def __init__(self, r):
        self.r = r
    def area(self): return Circle.PI * self.r * self.r
    def circumference(self): return 2 * Circle.PI * self.r
    @classmethod
    def set_pi(cls, val): cls.PI = val; print('PI set to', val)
    @staticmethod
    def valid_radius(r): return r>0
    def __del__(self): print('Circle destroyed')

circ = Circle(3)
print('Area:', round(circ.area(),2))
print('Valid?', Circle.valid_radius(3))
Circle.set_pi(3.14)
del circ
```

Output:

```
Error while running code:
Traceback (most recent call last):
  File "/tmp/ipykernel_11/998406400.py", line 13, in run_code_capture_output
    exec(code, globals_dict)
  File "<string>", line 1, in <module>
  File "<string>", line 2, in Circle
NameError: name 'math' is not defined
```

7. Book Details: title, author, price. display, update publisher (class), check price limit (static), destructor.

```
class Book:
    publisher = 'PubA'
    def __init__(self, title, author, price):
        self.title = title; self.author = author; self.price =
price
    def display(self): print(f'{self.title} by {self.author} -
{self.price}')
    @classmethod
    def set_publisher(cls, p): cls.publisher = p;
print('Publisher set to', p)
    @staticmethod
    def price_within_limit(price, limit): return price <= limit
    def __del__(self): print('Book destroyed')

b = Book('Python','XYZ',500)
b.display()
print('Within 600?', Book.price_within_limit(b.price,600))
Book.set_publisher('NewPub')
del b
```

Output:

```
Python by XYZ - 500
Within 600? True
Publisher set to NewPub
Book destroyed
```

8. Laptop Specifications: brand, model, ram. display, update warranty (class), check RAM sufficiency (static), destructor.

```
class Laptop:
    warranty_years = 1
    def __init__(self, brand, model, ram):
        self.brand = brand; self.model = model; self.ram = ram
    def specs(self): print(f'{self.brand} {self.model} -
{self.ram}GB, Warranty: {Laptop.warranty_years}yrs')
    @classmethod
    def set_warranty(cls, y): cls.warranty_years = y;
print('Warranty set to', y)
    @staticmethod
```

```
    def sufficient_ram(ram): return ram >= 8
    def __del__(self): print('Laptop destroyed')

lap = Laptop('Dell','XPS',16)
lap.specs()
print('Sufficient RAM?', Laptop.sufficient_ram(lap.ram))
Laptop.set_warranty(3)
del lap
```

Output:

```
Dell XPS - 16GB, Warranty: 1yrs
Sufficient RAM? True
Warranty set to 3
Laptop destroyed
```

9. Payroll System: name, basic_salary. total salary, update HRA% (class), calculate tax (static), destructor.

```
class Payroll:
    HRA_PERCENT = 20
    def __init__(self, name, basic):
        self.name = name; self.basic = basic
    def total_salary(self):
        hra = self.basic * Payroll.HRA_PERCENT/100
        return self.basic + hra
    @classmethod
    def set_hra(cls, p): cls.HRA_PERCENT = p; print('HRA set to',
p)
    @staticmethod
    def tax(amount, rate=10): return amount * rate/100
    def __del__(self): print('Payroll object destroyed')

p = Payroll('Sam',20000)
print('Total salary:', p.total_salary())
print('Tax on 30000:', Payroll.tax(30000,12))
Payroll.set_hra(25)
del p
```

Output:

```
Total salary: 24000.0
Tax on 30000: 3600.0
HRA set to 25
Payroll object destroyed
```

10. Temperature Converter: Celsius. convert to F, set boiling point (class), validate temp (static), destructor.

```
class Temperature:
    boiling_point = 100
    def __init__(self, c):
        self.c = c
    def to_fahrenheit(self): return (self.c * 9/5) + 32
    @classmethod
    def set_boiling(cls, b): cls.boiling_point = b;
print('Boiling point set to', b)
    @staticmethod
    def valid_temp(c): return -273.15 <= c
    def __del__(self): print('Temperature object destroyed')

t = Temperature(25)
print('F:', t.to_fahrenheit())
print('Valid?', Temperature.valid_temp(25))
Temperature.set_boiling(150)
del t
```

Output:

```
F: 77.0
Valid? True
Boiling point set to 150
Temperature object destroyed
```

11. Product Management: name, price, quantity. total cost, update tax rate (class), validate quantity (static), destructor.

```
class Product:
    tax_rate = 0.05
    def __init__(self, name, price, qty):
        self.name = name; self.price = price; self.qty = qty
    def total_cost(self):
        return self.price * self.qty * (1 + Product.tax_rate)
    @classmethod
    def set_tax(cls, r): cls.tax_rate = r; print('Tax set to', r)
    @staticmethod
    def valid_qty(q): return q >= 0
    def __del__(self): print('Product destroyed')

prod = Product('Pen',10,5)
print('Total cost:', prod.total_cost())
print('Valid qty?', Product.valid_qty(prod.qty))
Product.set_tax(0.1)
del prod
```

Output:

```
Total cost: 52.5
Valid qty? True
Tax set to 0.1
Product destroyed
```

12. Employee Attendance: emp_name, days_present. display attendance, update total working days (class), check bonus eligibility (static), destructor.

```
class Attendance:
    total_working = 22
    def __init__(self, emp, days):
        self.emp = emp; self.days = days
    def display(self): print(f'{self.emp} present
{self.days}/{Attendance.total_working}')
    @classmethod
    def set_total(cls, t): cls.total_working = t; print('Total
```

```
working days set to', t)
    @staticmethod
    def bonus_eligible(days): return days >= 20
    def __del__(self): print('Attendance object destroyed')

a = Attendance('Rita',21)
a.display()
print('Eligible?', Attendance.bonus_eligible(a.days))
Attendance.set_total(24)
del a
```

Output:

```
Rita present 21/22
Eligible? True
Total working days set to 24
Attendance object destroyed
```

13. CircleMath Operations: radius. area/circumference, update π (class), compare circle areas (static), destructor.

```
class CircleMath:
    PI = math.pi
    def __init__(self, r): self.r = r
    def area(self): return CircleMath.PI * self.r * self.r
    def circumference(self): return 2 * CircleMath.PI * self.r
    @classmethod
    def set_pi(cls, v): cls.PI = v; print('PI set to', v)
    @staticmethod
    def compare(c1, c2):
        a1 = c1.area(); a2 = c2.area()
        if a1>a2: return 'c1 bigger'
        if a1<a2: return 'c2 bigger'
        return 'equal'
    def __del__(self): print('CircleMath destroyed')

c1 = CircleMath(3); c2 = CircleMath(4)
print('Areas:', round(c1.area(),2), round(c2.area(),2))
print('Compare:', CircleMath.compare(c1,c2))
CircleMath.set_pi(3.1416)
```

```
del c1; del c2
```

Output:

```
Error while running code:
Traceback (most recent call last):
  File "/tmp/ipykernel_11/998406400.py", line 13, in run_code_capture_output
    exec(code, globals_dict)
  File "<string>", line 1, in <module>
  File "<string>", line 2, in CircleMath
NameError: name 'math' is not defined
```

14. Student Marks: name and marks. total & average, update passing marks (class), assign grade (static), destructor.

```
class StudentMarks:
    passing = 40
    def __init__(self, name, marks):
        self.name = name; self.marks = marks
    def total(self): return sum(self.marks)
    def average(self): return self.total()/len(self.marks)
    @classmethod
    def set_passing(cls, p): cls.passing = p; print('Passing set
to', p)
    @staticmethod
    def assign_grade(avg):
        if avg>=90: return 'A+'
        if avg>=75: return 'A'
        if avg>=50: return 'B'
        return 'F'
    def __del__(self): print('StudentMarks destroyed')

s = StudentMarks('Tony',[60,70,80])
print('Total:', s.total(), 'Avg:', s.average())
print('Grade:', StudentMarks.assign_grade(s.average()))
StudentMarks.set_passing(45)
del s
```

Output:

```
Total: 210 Avg: 70.0
Grade: B
Passing set to 45
StudentMarks destroyed
```

15. Vehicle Information: type, brand, speed. display, update total vehicle count (class), check legal speed (static), destructor.

```
class Vehicle:
    total = 0
    def __init__(self, vtype, brand, speed):
        self.vtype = vtype; self.brand = brand; self.speed =
speed
        Vehicle.total += 1
    def display(self): print(f'{self.vtype} {self.brand} @
{self.speed} km/h')
    @classmethod
    def set_total(cls, t): cls.total = t; print('Total vehicles
set to', t)
    @staticmethod
    def legal_speed(speed, limit=60): return speed <= limit
    def __del__(self): print('Vehicle destroyed')

v = Vehicle('Car','Toyota',80)
v.display()
print('Legal?', Vehicle.legal_speed(v.speed,90))
Vehicle.set_total(100)
del v
```

Output:

```
Car Toyota @ 80 km/h
Legal? True
Total vehicles set to 100
Vehicle destroyed
```