ICP 2 REPORT

```python
class Employee:
    # Class variable to keep track of the number of employees
    employee_count = 0

    def __init__(self, name, family, salary, department):
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        # Increment the employee count when a new instance is created
        Employee.employee_count += 1

    @staticmethod
    def average_salary(employees):
        if not employees:
            return 0
        total_salary = sum(emp.salary for emp in employees)
        return total_salary / len(employees)

    def __str__(self):
        return f"Employee(Name: {self.name}, Family: {self.family}, Salary: {self.salary}, Department: {self.department})"

class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department, benefits):
        super().__init__(name, family, salary, department)
        self.benefits = benefits
```

```python
        self.benefits = benefits

    def __str__(self):
        return f"FulltimeEmployee(Name: {self.name}, Family: {self.family}, Salary: {self.salary}, Department: {self.department}, Benefits: {self.benefits

# Creating instances of Employee and FulltimeEmployee
emp1 = Employee("sachin", "tendulkar", 100000, "chief executor")
emp2 = Employee("dhoni", "singh", 70000, "manager")
emp3 = FulltimeEmployee("rohit", "sharma", 120000, "accounts", "discounts on entertainment ")

# List of employees to calculate average salary
employees = [emp1, emp2, emp3]

# Print details of each employee
print(emp1)
print(emp2)
print(emp3)

# Calculate and print average salary
average_salary = Employee.average_salary(employees)
print(f"Average Salary: {average_salary}")

# Print the total number of employees
print(f"Total Number of Employees: {Employee.employee_count}")
```

```
Employee(Name: sachin, Family: tendulkar, Salary: 100000, Department: chief executor)
Employee(Name: dhoni, Family: singh, Salary: 70000, Department: manager)
FulltimeEmployee(Name: rohit, Family: sharma, Salary: 120000, Department: accounts, Benefits: discounts on entertainment )
Average Salary: 96666.66666666667
Total Number of Employees: 3
```

1. The difference between counter.count and self._count is :

Counter.count:

1. Shared across all instances.
2. Tracks the total number of increment calls across all instances.
3. keeps track of a value common to all instances.
4. Can be accessed by using counter

• self._count:

1. Unique to each instance.
2. Tracks the number of increment calls for that specific instance.
3. Can be accessed by using self with instance methods.

2. The outputs are: a.get_counts() will print: "Instance count: 2, Class count: 3" b.get_counts() will print: "Instance count: 1,class count :3"
3. How does the increment method affect both the class and instance variables?

   The increment method affects both the class variable count and the instance variable _count:

Increment method on Class Variable (Counter.count) When the increment method is called, the line Counter.count += 1 increases the value of this class variable by 1. This means that every time increment is called on any instance, Counter.count will be incremented, and this change is reflected across all instance.

Increment method on Instance Variable (self._count) The instance variable `_count` is unique to each instance of the Counter class. When the increment method is called, the line self._count += 1 increases the value of this instance variable by 1 for that specific instance. This means that the increment only affects the _count of the instance on which the method is called.

My Github link is : https://github.com/PraveenDondapati/bda .git