In [31]:
```
# LIBRARIES
import random
import time
from datetime import datetime
import pandas as pd
from abc import ABC,abstractmethod
from collections import deque
```

In [3]:
```
# PYTHON COLOR CODES TO PRINT FONT OF OWN COLOR CHOICE

color  = {
            'PURPLE' : '\033[95m',
            'CYAN' : '\033[96m',
            'DARKCYAN' : '\033[36m',
            'BLUE' : '\033[94m',
            'GREEN' : '\033[92m',
            'YELLOW' : '\033[93m',
            'RED' : '\033[91m',
            'BOLD' : '\033[1m',
            'UNDERLINE' : '\033[4m',
            'END' : '\033[0m'
        }
```

In [70]:
```
# FUNCTION TO DISPLAY PROCESS GOING ON

def process(msg):
     print("\n")
     t = random.randint(2,6)
     for i in range(t+1):
        info = '----->> {0:^20} <<-----'.format(msg)
        print_colored(info,"BLUE")
        time.sleep(1)
     print("\n")
```

In [4]:
```
# FUNCTION TO PRINT COLORED TEXT
def print_colored(message,msg_color=None):
       print((color['BOLD']+color[msg_color] + message +color['END']).center(125))

# FUNCTION TO DISPLAY MESSAGE WITH CERTAIN COLOR
def display(message,color):
     print()
     pattern = "**********************************************************************"
     print_colored(pattern,color)
     print_colored(message,color)
     print_colored(pattern,color)
     print()

# TESTING
display("Hey there! How are you",'PURPLE')
```

```
**********************************************************************
                        Hey there! How are you
**********************************************************************
```

In [7]:
```
class Vehicle(ABC):
    """
    An abstract class to force its subclass to implement abstract attribute and abstract method
    """

    # ABSTRACT ATTRIBUTE
    @property
    @abstractmethod
    def vehicle_type(self) -> str:
          raise NotImplementedError

    # ABSTRACT METHOD
    @abstractmethod
    def __str__(self) -> str:
          raise NotImplementedError
```

In [11]:
```
# Models being used in the inventory
Car_Models = {"Maruti Alto":1500,"Maruti Swift":2000,"Hyundai Creta":2500,"Mahindra Thar":3000,"Toyota Fortun
er":4000}
Bike_Models = {"Suzuki Access 125":300,"Honda Glammer":500,"Royal Enfield Bullet":1000}
```

In [15]:
```
class Car(Vehicle):
      """
      Car class inheriting abstract class Vehicle
      """
      vehicle_type = "Car"

      def __init__(self,number,model):
          """
          Constructor to initializes the bike object with field below
          """
          self.vehicle_number = number
          self.model = model
          self.rent_price = Car_Models[model]

      def __str__(self):
            """
            Format to print car
            """
            return f'Type : {Car.vehicle_type}\tId : {self.vehicle_number}\nModel : {self.model}\tRent_Price
 : {self.rent_price}'
```

In [54]:
```
class Bike(Vehicle):
      """
      Bike class inheriting abstract class Vehicle
      """
      vehicle_type = "Bike"

      def __init__(self,number,model):
          """
          Constructor to initializes the bike object with field below
          """
          self.vehicle_number = number
          self.model = model
          self.rent_price = Bike_Models[model]

      def __str__(self):
            """
            Format to print bike
            """
            return f'Type : {Bike.vehicle_type}\tId : {self.vehicle_number}\nModel : {self.model}\tRent_Price
: {self.rent_price}'
```

In [55]:
```
# FUNCTION TO DISPLAY BILL

def display_bill(customer,color):
    bill_amount = 0
    days_used = (datetime.now()-customer.rental_time).days
    print_colored("CUSTOMER INVOICE",color)
    pattern = "*********************************************************************************"
    print_colored(pattern,color)
    columns_info = '| {0:^12} | {1:^20} | {2:^12} | {3:^12} |'.format("VEHICLE","MODEL","PRICE/DAY","TOTAL P
RICE")
    print_colored(columns_info,color)
    print_colored(pattern,color)
    for vehicle in customer.vehicles:
        vehicle_info = '| {0:^12} | {1:^20} | {2:^12} | {3:^12} |'.format(vehicle.vehicle_type,vehicle.model
,vehicle.rent_price,vehicle.rent_price*days_used)
        print_colored(vehicle_info,color)
        bill_amount += (vehicle.rent_price)*days_used
    print_colored(pattern,color)
    print()
    return bill_amount
```

In [107]:
```
class Inventory:
      """
      Inventory class to contain list of bikes and cars
      """
      def __init__(self):
        """
        Constructor to initialize the inventory object with emmpty sets of vehicles
        """
        # set is used here so, adding and removing complexity is O(1)
        self.bikes = set()
        self.cars  = set()

      def add_item(self,item):
          """
          Add vehicle item in inventory
          """
          if item.vehicle_type == "Bike":
             self.bikes.add(item)
          else:
             self.cars.add(item)

      def del_item(self,item):
          """
          Remove vehicle item from inventory
          """
          if item.vehicle_type == "Bike":
             self.bikes.remove(item)
          else:
             self.cars.remove(item)

      def display_stocks(self):
          """
          Display the stocks in the inventory
          """
          display(f"We currently have {len(self.bikes)} bikes and {len(self.cars)} cars","PURPLE")

      def rent(self,customer,num_of_bikes=0,num_of_cars=0):
          """
          Rent the vehicles from inventory as requested
          """
          total_bikes = len(self.bikes)
          total_cars  = len(self.cars)
```

```
            process_status = False
            if   total_bikes <= 0 and total_cars <= 0:
                 display("Sorry !, Our Inventory is empty")

            elif num_of_bikes > total_bikes:
                 display("Sorry! We have currently only {} bikes available to rent.".format(total_bikes),"PURPL
E")

            elif num_of_cars > total_cars:
                 display("Sorry! We have currently only {} cars available to rent.".format(total_cars),"PURPLE"
)

            else :
                 # Replace """ datetime(2021,3,10) with datetime.now() """  in production mode
                 customer.rental_time = datetime(2021,3,10)

                 # Iterating over copy bcs the set is changed in the iteration
                 for bike in self.bikes.copy():
                       if num_of_bikes <= 0 :
                           break
                       num_of_bikes -= 1
                       self.del_item(bike)
                       customer.vehicles.append(bike)

                 for car in self.cars.copy():
                       if num_of_cars <= 0 :
                           break
                       num_of_cars -= 1
                       self.del_item(car)
                       customer.vehicles.append(car)
                 display("Request completed successully !","CYAN")
                 display("Enjoy the ride , Sir !","BLUE")
                 process_status = True

            return process_status

       def generate_bill(self,customer):
           """
           Display the bill and return the bill
           """
           bill_amt = display_bill(customer,"PURPLE")
           return bill_amt
```

In [108]:
```
# Inventory object
shop = Inventory()

# Adding items in the inventory
"""
In case we can read from file or fetch items from database to make it more real , for
simplicity we are adding manually
"""
Cars=[]
Bikes=[]
for model in Car_Models:
    for i in range(5):
        Cars.append(Car("C"+str(i)+model[0].upper()+model[-1].upper(),model))

for model in Bike_Models:
    for i in range(5):
        Bikes.append(Bike("B"+str(i)+model[0].upper()+model[-1].upper(),model))

random.shuffle(Cars)
random.shuffle(Bikes)

for car in Cars:
    shop.add_item(car)
for bike in Bikes:
    shop.add_item(bike)
```

In [109]:
```
# Testing purpose to see items in the inventory
for car in shop.cars:
    print(car,"\n")

for bike in shop.bikes:
    print(bike,"\n")
```

```
Type : Car      Id : C1TR
Model : Toyota Fortuner Rent_Price : 4000

Type : Car      Id : C4HA
Model : Hyundai Creta   Rent_Price : 2500

Type : Car      Id : C0TR
Model : Toyota Fortuner Rent_Price : 4000

Type : Car      Id : C0MO
Model : Maruti Alto     Rent_Price : 1500

Type : Car      Id : C3MR
Model : Mahindra Thar   Rent_Price : 3000

Type : Car      Id : C0HA
Model : Hyundai Creta   Rent_Price : 2500

Type : Car      Id : C3HA
Model : Hyundai Creta   Rent_Price : 2500

Type : Car      Id : C4MO
Model : Maruti Alto     Rent_Price : 1500

Type : Car      Id : C2HA
Model : Hyundai Creta   Rent_Price : 2500

Type : Car      Id : C3MO
Model : Maruti Alto     Rent_Price : 1500

Type : Car      Id : C1MO
Model : Maruti Alto     Rent_Price : 1500

Type : Car      Id : C3MT
Model : Maruti Swift    Rent_Price : 2000

Type : Car      Id : C4MT
Model : Maruti Swift    Rent_Price : 2000

Type : Car      Id : C2TR
Model : Toyota Fortuner Rent_Price : 4000

Type : Car      Id : C1MR
```

```
Model : Mahindra Thar   Rent_Price : 3000

Type : Car      Id : C0MT
Model : Maruti Swift    Rent_Price : 2000

Type : Car      Id : C1HA
Model : Hyundai Creta   Rent_Price : 2500

Type : Car      Id : C0MR
Model : Mahindra Thar   Rent_Price : 3000

Type : Car      Id : C2MR
Model : Mahindra Thar   Rent_Price : 3000

Type : Car      Id : C3TR
Model : Toyota Fortuner Rent_Price : 4000

Type : Car      Id : C4MR
Model : Mahindra Thar   Rent_Price : 3000

Type : Car      Id : C2MO
Model : Maruti Alto     Rent_Price : 1500

Type : Car      Id : C2MT
Model : Maruti Swift    Rent_Price : 2000

Type : Car      Id : C4TR
Model : Toyota Fortuner Rent_Price : 4000

Type : Car      Id : C1MT
Model : Maruti Swift    Rent_Price : 2000

Type : Bike     Id : B3S5
Model : Suzuki Access 125       Rent_Price : 300

Type : Bike     Id : B1RT
Model : Royal Enfield Bullet    Rent_Price : 1000

Type : Bike     Id : B1S5
Model : Suzuki Access 125       Rent_Price : 300

Type : Bike     Id : B1HR
Model : Honda Glammer   Rent_Price : 500
```

```
Type : Bike     Id : B4S5
Model : Suzuki Access 125       Rent_Price : 300

Type : Bike     Id : B2HR
Model : Honda Glammer   Rent_Price : 500

Type : Bike     Id : B3RT
Model : Royal Enfield Bullet    Rent_Price : 1000

Type : Bike     Id : B2S5
Model : Suzuki Access 125       Rent_Price : 300

Type : Bike     Id : B0HR
Model : Honda Glammer   Rent_Price : 500

Type : Bike     Id : B3HR
Model : Honda Glammer   Rent_Price : 500

Type : Bike     Id : B4HR
Model : Honda Glammer   Rent_Price : 500

Type : Bike     Id : B4RT
Model : Royal Enfield Bullet    Rent_Price : 1000

Type : Bike     Id : B0S5
Model : Suzuki Access 125       Rent_Price : 300

Type : Bike     Id : B2RT
Model : Royal Enfield Bullet    Rent_Price : 1000

Type : Bike     Id : B0RT
Model : Royal Enfield Bullet    Rent_Price : 1000
```

In [115]:
```
class Customer:
    def __init__(self,name,aadhar_number):
        """
        Our constructor method which initializes the customer object with field below
        """
        self.name = name
        self.aadhar_number = aadhar_number
        self.vehicles = deque()
        self.rental_time = 0
        self.bill = 0

    def request_vehicle(self):
        """
        Takes a request from the customer for the vehicles to be rented.
        """
        while True:
            num_of_bikes = input("Enter number of bikes to be rented ")
            num_of_cars  = input("Enter number of cars to be rented  ")
            try:
                num_of_bikes,num_of_cars  = int(num_of_bikes),int(num_of_cars)

            except ValueError:
                display("Please enter a integer value","RED")
                continue

            if num_of_bikes < 1 or num_of_cars < 1:
               display("Invalid input , number of vehicles should be greater than zero","RED")
            else:
                break

        display( f'Customer requested for {num_of_bikes} bikes and {num_of_cars} cars','GREEN')
        process("PROCESS GOING ON")
        # Funtion to rent the vehicles requested
        if shop.rent(self,num_of_bikes,num_of_cars) == False:
           self.request_vehicle()



    def return_vehicles(self):
        """
        Allows customers to return their vehicles to the rental shop after generating the bill.
        """
```

```
        display("Customer wants to return vehicles","GREEN")
        process("PROCESS GOING ON")
        self.bill = shop.generate_bill(self)
        display(f" Total Amount =  Rs. {self.bill}","PURPLE")
        for vehicle in self.vehicles.copy():
            if vehicle.vehicle_type == "Bike":
                shop.bikes.add(vehicle)
            else:
                shop.cars.add(vehicle)

            self.vehicles.remove(vehicle)
        display("Returned vehicles successfully !","CYAN")


    def pay_bill(self):
        """
        Pays the bill
        """
        display(f"Payment of Amt Rs. {self.bill} initiated","CYAN")
        process("Payment on process,wait for seconds")
        display(f"{self.name} with total bill of Rs.{self.bill} has paid successfully","GREEN")
        self.bill  = 0
        display("Thank You Sir, for having service from us!","BLUE")

    def __str__(self):
        """
        Format to print the customer
        """
        return f'Name : {self.name}\tAadhar Number: {self.aadhar_number}'
```

In [116]:
```
# Customer Object
c = Customer(name = "James Bond",aadhar_number = "A01SF89234023")
```

In [117]:
```
# Display the stocks in the shop
shop.display_stocks()
# Customer request for vehicle
c.request_vehicle()
```

****
We currently have 15 bikes and 25 cars
****

```
Enter number of bikes to be rented asdf
Enter number of cars to be rented  as
```

****
Please enter a integer value
****

```
Enter number of bikes to be rented -1
Enter number of cars to be rented  0
```

****
Invalid input , number of vehicles should be greater than zero
****

```
Enter number of bikes to be rented 3
Enter number of cars to be rented  2
```

****
Customer requested for 3 bikes and 2 cars
****

----->> PROCESS GOING ON <<-----
----->> PROCESS GOING ON <<-----
----->> PROCESS GOING ON <<-----
----->> PROCESS GOING ON <<-----
----->> PROCESS GOING ON <<-----
----->> PROCESS GOING ON <<-----

****
Request completed successully !
****

****
Enjoy the ride , Sir !

****

In [118]:
```
# Customer returns the veihcle
c.return_vehicles()
```

```
**********************************************************************
                    Customer wants to return vehicles
**********************************************************************



                ----->>   PROCESS GOING ON   <<-----
                ----->>   PROCESS GOING ON   <<-----
                ----->>   PROCESS GOING ON   <<-----
                ----->>   PROCESS GOING ON   <<-----
                ----->>   PROCESS GOING ON   <<-----
                ----->>   PROCESS GOING ON   <<-----
                ----->>   PROCESS GOING ON   <<-----


                              CUSTOMER INVOICE
**************************************************************************************
     |   VEHICLE    |        MODEL         |  PRICE/DAY   | TOTAL PRICE  |
**************************************************************************************
     |     Bike     |  Suzuki Access 125   |     300      |     1200     |
     |     Bike     | Royal Enfield Bullet |     1000     |     4000     |
     |     Bike     | Royal Enfield Bullet |     1000     |     4000     |
     |     Car      |     Maruti Alto      |     1500     |     6000     |
     |     Car      |     Maruti Alto      |     1500     |     6000     |
**************************************************************************************


     **********************************************************************
                           Total Amount =  Rs. 21200
     **********************************************************************


     **********************************************************************
                        Returned vehicles successfully !
     **********************************************************************
```

In [119]:
```
# Customer pays the bill
c.pay_bill()
```

```
**********************************************************************
                    Payment of Amt Rs. 21200 initiated
**********************************************************************



          ----->> Payment on process,wait for seconds <<-----
          ----->> Payment on process,wait for seconds <<-----
          ----->> Payment on process,wait for seconds <<-----
          ----->> Payment on process,wait for seconds <<-----
          ----->> Payment on process,wait for seconds <<-----
          ----->> Payment on process,wait for seconds <<-----
          ----->> Payment on process,wait for seconds <<-----


**********************************************************************
    James Bond with total bill of Rs.21200 has paid successfully
**********************************************************************


**********************************************************************
              Thank You Sir, for having service from us!
**********************************************************************
```