Project Topic:
Designing a Parking System for Kigali Arena Sporting Events

Class Files

```
class Vehicle:
  """ This class holds all our vehicles, checks the time and license number"""
  def __init__(self):
    self.license_number = uuid.uuid1()
    self.time = time.time()


class Car(Vehicle):
  """ Car class representing one of the vehicles we have in our vehicle class """
  def __init__(self):
    Vehicle.__init__(self)

class Moto(Vehicle):
  "Moto class representing one of the vehicles we have in our vehicle class "
  def __init__(self):
    Vehicle.__init__(self)


class Truck(Vehicle):
  "Truck class representing one of the vehicles we have in our vehicle class "
  def __init__(self):
    Vehicle.__init__(self)
```

DOCUMENTATION

As explained above the vehicle class contains all the information about the vehicles and allows to determine the time and the registration number, we know cars, as well as motorbikes all, have a registration number plate so the vehicle class will allow determining the vehicle number (car, motorbike, truck) and the time to enter or exit the vehicles of which the class Car, Moto, and Truck will be an inheritance of the vehicle class.

```
class ParkingSlot:
  def __init__(self, name):
    self.name = name
    self.capacity = ???
    self.time = []
```

```
        self.lots = []

"This method adds vehicles in our parking slots"

  def add_vehicle(self, vehicle):
        if len(self.lots) < self.capacity:
            self.lots.append(vehicle)
            """ This method helps us view details for vehicle parked and serves as accountability
measure"""

 def view_details(self):
        for vehicle in self.lots:


    "The subclasses inheriting from the Parking Slot"


class CarParkingSlot(ParkingSlot):
    def __init__(self, name):
        ParkingSlot.__init__(self, name)
        self.capacity = ?????

The role of this Class is to direct the cars towards the car parking and to indicate if there are any
free places.

class MotoParkingSlot(ParkingSlot):
    def __init__(self, name):
        ParkingSlot.__init__(self, name)
        self.capacity = ?????

The role of this Class is to direct the moto towards the moto parking and to indicate if there are
any free places.

class TruckParkingSlot(ParkingSlot):
    def __init__(self, name):
        ParkingSlot.__init__(self, name)
        self.capacity = ????

The role of this Class is to direct the Trucks towards the car parking and to indicate if there are
any free places.
```

DOCUMENTATION

The class parking slot is considered as the parent class of all other class parking slots (car, motorbike, trucks) it helps us to visualize the details of the car and add the cars. The class, car parking slot, Moto Parking slots, and Truck Parking slots are daughter class (inheritance) will have the same functions as the class Parking slots with the only difference that each one adds and will give specific information, for example, the class Truck Parking slots only gives and will only add information about Trucks.

Class **Controller :**

```
def __init__(self):
   pass
```

"The static method here runs the method without instantiating it with the controller class"

```
def control_vehicles(vehicle_type):
```

This method controls the vehicle type and the appropriate parking lot for it

DOCUMENTATION:

The controller class determines the type of vehicle and sends it to the correct car park.
This class will be linked to all the other classes because it uses the information generated by the different classes to determine the type of vehicle and will indicate whether it has parking spaces.

CLASS TEST

1-TEST DOCUMENTATION:

For testing our program we will use the Unittest
The unittest module is imported for this purpose.
The purpose here is to determine whether our class is working.
We will further support our tests on the car park and the different type of vehicles it will allow to show at the same time if our inheritance and methods work.

TEST TABLE

Class vehicles

Test cas number	Test car description	Test data	Expected result	Actual result	pass/fail
1	We will test the vehicle class and see if the class transmits the correct information (time, registration number).	Module time and	the program gives the exact time to exit and enter the vehicle with matriculation	the program gives the exact time and the matriculation number	pass
2	We will test the Car class and see if the class transmits the correct information (time).	Modul time	the program gives the exact time to exit and enter the car	the program gives the exact time	pass
3	We will test the Moto class and see	Modul time	the program gives the exact time to	the program gives the exact time	pass

	if the class transmits the correct information (time).		exit and enter the Moto		
4	We will test the Trucks class and see if the class transmits the correct information (time).	Modul time	the program gives the exact time to exit and enter the Truck	the program gives the exact time	pass
5	Test if Truck class is an instance of Vehicle	Module instances	If the program can give an example	The program gives and shows an example	pass

Class Parking slot

Test case number	Test car description	Test data	Expected result	Actual result	pass/fail
1	Test the capacity of the parking	Capacity (30)	That the program can show the parking's capacity	The program gives the capacity	pass
2	we test if the function for adding the vehicles works	All vehicle	That the program adding the news cars	The program shows the new vehicle in the parking	pass

3	Test the default duration of the parking slot	Module time and vehicle	That the program shows the parked time of each vehicle	The program works and shows the time	pass
4	We test the capacity of the parking slots at the departure before the arrival of the car.	Parking initial capacity,	That the program gives the capacity of the parking and says the place is available or not after the depart of a car or arrival	The program works and gives the capacity	pass
5	We test if the cars find parking spaces.	That the program can see if the parking have the spaces or not and can give a space has a car or motorbike		The program work and shows the available space	pass

UNITTEST CLASS CODE

```
# ------- Vehicle Parking System for Kigali Arena ------- #

# @Authors : Achille Tanwouo and Samuel Anumudu

# Test our program with Unittest Module
import unittest
from unittest import TestCase
from car_class import Car
```

```
from controller import Controller
from moto_class import Moto
from parking import CarParkingSlot, ParkingSlot
from truck_class import Truck


# Test Moto Class datatype of time ---------       TEST 1
class TestMoto(TestCase):
  def test_time_for_moto(self):
     moto_1 = Moto()
     self.assertEqual(type(moto_1.time), float)


# Test Car Class datatype of time ---------       TEST 2
class TestCar(TestCase):
  def test_time_for_car(self):
     car_1 = Car()
     self.assertEqual(type(car_1.time), float)


# Test Truck Class datatype of time ---------       TEST 3
class TestTruck(TestCase):
  def test_time_for_truck(self):
     truck_1 = Truck()
     self.assertEqual(type(truck_1.time), float)


# Test if Parking Slot capacity is 30 for all types vehicles--------      TEST 4
class TestParkingSlot(TestCase):
  def test_capacity(self):
     car_parking_slot_1 = CarParkingSlot('A')
     self.assertEqual(car_parking_slot_1.capacity, 30)


# Test default length of ParkingSlot ----------     TEST 5
class TestParkingSlot(TestCase):
  def test_initial_parkingSlot(self):
     parkingSlot = ParkingSlot("ParkingSlot_A")
     self.assertEqual(len(parkingSlot.lots), 0)
```

```
  def test_add_vehicle(self):
    parkingSlot_1 = ParkingSlot("A")
    parkingSlot_1.add_vehicle(Moto())
    parkingSlot_1.add_vehicle(Moto())
    parkingSlot_1.add_vehicle(Moto())
    self.assertEqual(len(parkingSlot_1.lots), 3)

  def test_view_details(self):
    parkingSlot_1 = ParkingSlot('A')
    parkingSlot_1.lots
    self.assertEqual(type(parkingSlot_1.lots), list)


# Test if Car class is an instance of Vehicle -------- TEST 6
class TestCar(TestCase):
  def test_is_instance(self):
    self.assertIsInstance(Car(), Car)


# Test if Truck class is an instance of Vehicle -------- TEST 7
class TestTruck(TestCase):
  def test_is_instance(self):
    self.assertIsInstance(Truck(), Truck)


# Test if Truck class is an instance of Vehicle -------- TEST 8
class TestMoto(TestCase):
  def test_is_instance(self):
    self.assertIsInstance(Moto(), Moto)


# This test assert True that Controller is a class-object ----- TEST 9
class TestController(TestCase):
  def test_class(self):
    controller_class = Controller()
    self.assertTrue(type(controller_class), True)


if __name__ == '__main__':
  unittest.main()
```