#car.py

class Car:  
 *"""Represent a Car object."""* def \_\_init\_\_(self, name="", fuel=0):  
 *"""Initialise a Car instance.  
 fuel: float, one unit of fuel drives one kilometre  
 """* self.name = name  
 self.fuel = fuel  
 self.odometer = 0  
  
 def \_\_str\_\_(self):  
 return "{}, fuel = {}, odometer ={}".format(self.name , self.fuel, self.odometer)  
  
 def add\_fuel(self, amount):  
 *"""Add amount to the car's fuel."""* self.fuel += amount  
  
 def drive(self, distance):  
 *"""Drive the car a given distance.  
 Drive given distance if car has enough fuel  
 or drive until fuel runs out return the distance actually driven.  
 """* if distance > self.fuel:  
 distance = self.fuel  
 self.fuel = 0  
 else:  
 self.fuel -= distance  
 self.odometer += distance  
 return distance

#taxi.py  
from car import Car  
  
  
class Taxi(Car):  
 *"""Specialised version of a Car that includes fare costs."""* price\_per\_km = 1.23  
  
 def \_\_init\_\_(self, name, fuel, ):  
 *"""Initialise a Taxi instance, based on parent class Car."""* super().\_\_init\_\_(name, fuel)  
  
 self.current\_fare\_distance = 0  
 self.odometer = 0  
  
 def \_\_str\_\_(self):  
 *"""Return a string like a Car but with current fare distance."""* return "{}, {}km on current fare, ${:.2f}/km".format(super().\_\_str\_\_(),  
 self.current\_fare\_distance,  
 self.price\_per\_km)  
  
 def get\_fare(self):  
 *"""Return the price for the taxi trip."""* return round(self.price\_per\_km \* self.current\_fare\_distance, 1)  
  
 def start\_fare(self):  
 *"""Begin a new fare."""* self.current\_fare\_distance = 0  
  
 def drive(self, distance):  
 *"""Drive like parent Car but calculate fare distance as well."""* distance\_driven = super().drive(distance)  
 self.current\_fare\_distance += distance\_driven  
 return distance\_driven

#taxi.simulator.py

from taxi import Taxi  
from SilverServiceTaxi import SilverServiceTaxi  
  
  
def main():  
 *"""Drive car program"""* taxis = [Taxi("Prius", 100), SilverServiceTaxi("Limo", 100, 2), SilverServiceTaxi("Hummer", 200, 4)]  
 bill\_to\_date = 0  
 current\_taxi = None  
 while True:  
 print('Let\'s drive!')  
 print('(C)hoose Taxi')  
 print('(D)rive')  
 print('(Q)uit')  
 choice = input('>>>').upper()  
 if choice == 'C':  
 for index, taxi in enumerate(taxis):  
 print('{} - {}'.format(index, taxi))  
  
 current\_taxi = int(input('Choose taxi: '))  
 bill\_to\_date += taxis[current\_taxi].get\_fare()  
 print('Bill to date: ', round(bill\_to\_date, 1))  
 elif choice == 'D':  
 if current\_taxi is not None:  
 distance = int(input('Drive how far? '))  
 taxis[current\_taxi].drive(distance)  
 print('Your {} trip cost you {}'.format(taxis[current\_taxi].name, taxis[current\_taxi].get\_fare()))  
 bill\_to\_date += taxis[current\_taxi].get\_fare()  
 print('Bill to date: ', round(bill\_to\_date, 1))  
 else:  
 print('Please choose a taxi first!')  
 pass  
 elif choice == 'Q':  
 print('Total trip cost: ', round(bill\_to\_date, 1))  
 quit()  
 else:  
 print('Invalid menu choice')  
  
  
main()

#taxi\_test.py

from taxi import Taxi  
  
  
def main():  
 taxi = Taxi("Prius 1", 100)  
 taxi.drive(40)  
 print(taxi.\_\_str\_\_())  
 print("Taxi's current fare: {}".format(taxi.get\_fare()))  
 taxi.start\_fare()  
 taxi.drive(100)  
 print("Taxi detail - Name: {}, fuel: {}, price per km: {}".format(taxi.name, taxi.fuel, taxi.price\_per\_km))  
 print("Taxi's current fare: {}".format(taxi.get\_fare()))  
  
  
main()

#SilverServiceTaxi.py

from taxi import Taxi  
  
  
class SilverServiceTaxi(Taxi):  
 flagfall = 4.50  
  
 def \_\_init\_\_(self, name, fuel, fanciness):  
 super().\_\_init\_\_(name, fuel)  
 self.fanciness = fanciness  
 self.price\_per\_km = self.price\_per\_km \* fanciness  
  
 def \_\_str\_\_(self):  
 return "{} plus flagfall of ${}".format(super().\_\_str\_\_(), self.flagfall)  
  
 def get\_fare(self):  
 return round(super().get\_fare() + self.flagfall if self.current\_fare\_distance != 0 else 0, 1)

#silver\_service\_taxi\_test.py

from SilverServiceTaxi import SilverServiceTaxi  
  
  
def main():  
 my\_silver\_taxi = SilverServiceTaxi("Silver\_Taxi\_1", 100, 2)  
 my\_silver\_taxi.drive(18)  
 print("Taxi's current fare: ${}".format(my\_silver\_taxi.get\_fare()))  
 print(my\_silver\_taxi.\_\_str\_\_())  
  
  
main()

#UnrealiableCar.py

from car import Car  
import random  
class UnreliableCar(Car):  
 price\_per\_km = 1.23  
 def \_\_init\_\_(self, name, fuel, reliability):  
 super().\_\_init\_\_(name, fuel)  
 self.reliability = reliability  
 self.current\_fare\_distance = 0  
 def \_\_str\_\_(self):  
 *"""Return a string ofa Car with the fare of distance."""* return "{}, {}km on current fare, ${:.2f}/km".format(super().\_\_str\_\_(),  
 self.current\_fare\_distance,  
 self.price\_per\_km)  
  
 def total\_fare(self):  
 *"""Return the price of the trip."""* return self.price\_per\_km \* self.current\_fare\_distance  
  
 def new\_fare(self):  
 *"""Start a new fare for driver."""* self.current\_fare\_distance = 0  
  
 def drive(self, distance):  
 distance\_driven = super().drive(distance)  
 if random.randint(1, 100) < self.reliability:  
 self.current\_fare\_distance += distance\_driven  
 else:  
 pass  
 return distance\_driven