Prac08：

taxi\_test.py

from taxi import Taxi  
  
def main():  
 taxi=Taxi(**'Prius 1'**, 100)  
 taxi.drive(40)  
 print(taxi)  
 print(**'fare = ${}'**.format(taxi.get\_fare()))  
 taxi.start\_fare()  
 taxi.drive(100)  
 print(taxi)  
 print(**'fare = ${}'**.format(taxi.get\_fare()))  
  
main()

taxi.py modifications (including class variable)

*"""  
CP1404/CP5632 Practical  
Car class  
"""*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  
  
 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 self.price\_per\_km \* self.current\_fare\_distance  
  
 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

unreliable\_car.py

from car import Car  
import random  
  
class UnreliableCar(Car):  
  
 def \_\_init\_\_(self, name, fuel, reliability):  
 super().\_\_init\_\_(name, fuel)  
 self.reliability = reliability  
  
 def drive(self, distance):  
 if random.randint(0, 100)<self.reliability:  
 return super().drive(distance)  
 else:  
 return super().drive(0)

unreliable\_car\_test.py

from unreliable\_car import UnreliableCar  
  
def main():  
 unreliablecar=UnreliableCar(**'car'**, 500, 50)  
 time=int(input(**'Enter the time you want to test: '**))  
 for i in range(time):  
 print(**'{} drives {}km'**.format(unreliablecar.name, unreliablecar.drive(10)))  
 print(unreliablecar)  
  
main()

silver\_service\_taxi.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 super().get\_fare() + self.flagfall

silver\_service\_taxi\_test.py

from silver\_service\_taxi import SilverServiceTaxi  
  
def main():  
 taxi=SilverServiceTaxi(**'taxi'**, 100, 2)  
 taxi.drive(18)  
 print(taxi)  
 print(**'fare = ${}'**.format(taxi.get\_fare()))  
  
main()

taxi\_simulator.py

（错的）

from taxi import Taxi  
from silver\_service\_taxi import SilverServiceTaxi  
  
  
def list\_taxi(taxis):  
 mark=0  
 for i in range(len(taxis)):  
 print(**'{} - {}'**.format(mark, taxis[i]))  
 mark+=1  
  
  
def choose\_taxi(taxis):  
 print(**"Taxis available: "**)  
 mark = 0  
 for i in range(len(taxis)):  
 print(**'{} - {}'**.format(mark, taxis[i]))  
 mark += 1  
  
 taxi\_choice=int(input(**"Choose taxi: "**))  
 if taxi\_choice<0 or taxi\_choice>len(taxis):  
 print(**'Invalid choice**\n**'**)  
  
  
def drive(taxis, current\_taxi, total\_cost):  
 if current\_taxi == None:  
 print(**'Please choose a taxi first!'**)  
 else:  
 current\_taxi.start\_fare()  
 distance=int(input(**"Drive how far? "**))  
 current\_taxi.drive(distance)  
 cost=current\_taxi.get\_fare()  
 print(**"Your {} trip cost you ${:.2f}"**.format(current\_taxi.name, cost))  
 total\_cost+=cost  
  
  
def main():  
 current\_taxi = None  
 total\_cost = 0  
 taxis = [Taxi(**"Prius"**, 100), SilverServiceTaxi(**"Limo"**, 100, 2), SilverServiceTaxi(**"Hummer"**, 200, 4)]  
 print(**'Let**\'**s drive!'**)  
 while True:  
 choice=str(input(**'q)uit, c)hoose taxi, d)rive**\n**>>> '**)).lower()  
  
 valid\_choice = **"qcd"** if choice not in valid\_choice:  
 print(**"Invalid choice.**\n**"**)  
  
 if choice == **'c'**:  
 current\_taxi=choose\_taxi(taxis)  
  
 if choice == **'d'**:  
 drive(taxis, current\_taxi, total\_cost)  
  
 if choice == **'q'**:  
 print(**"Total trip cost: ${:.2f}"**.format(total\_cost))  
 print(**"Taxis are now:"**)  
 list\_taxi(taxis)  
  
 print(**"Bill to date: ${:.2f}**\n**"**.format(total\_cost))  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

（对的）

from taxi import Taxi  
from silver\_service\_taxi import SilverServiceTaxi  
  
  
def get\_user\_select():valid\_selects = [**"q"**, **"c"**, **"d"**]  
 while True:  
 print(**"q)uit, c)hoose taxi, d)rive"**)  
 select = input(**">>> "**).lower()  
 if select in valid\_selects:  
 return select  
 else:  
 print(**"Invalid option"**)  
  
  
def show\_cars(cars):for i in range(len(cars)):  
 print(**"{} - {}"**.format(i, cars[i]))  
  
  
def get\_choose\_car(cars):while True:  
 print(**"Taxis available: "**)  
 show\_cars(cars)  
 car\_index = int(input(**"Choose taxi: "**))  
 if 0 <= car\_index < len(cars):  
 return cars[car\_index]  
 print(**"Invalid input"**)  
  
  
def main():  
 *# total bill* total\_cost = 0  
 *# Generate three cars* cars = [Taxi(**"Prius"**, 100), SilverServiceTaxi(**"Limo"**, 100, 2), SilverServiceTaxi(**"Hummer"**, 200, 4)]  
 *# Current operate car* cur\_car = None  
 print(**"Let's drive!"**)  
 user\_select = get\_user\_select()  
 while user\_select != **"q"**:  
 if user\_select == **"c"**:  
 *# Choose a car* cur\_car = get\_choose\_car(cars)  
 elif user\_select == **"d"**:  
 *# Drive a car* if cur\_car is None:  
 print(**"You should choose a taxi first!"**)  
 else:  
 cur\_car.start\_fare()  
 distance\_to\_drive = float(input(**"Drive how far? "**))  
 cur\_car.drive(distance\_to\_drive)  
 trip\_cost = cur\_car.get\_fare()  
 print(**"Your {} trip cost you ${:.2f}"**.format(cur\_car.name, trip\_cost))  
 total\_cost += trip\_cost  
 print(**"Bill to date: ${:.2f}"**.format(total\_cost))  
 user\_select = get\_user\_select()  
  
 print(**"Total trip cost: ${:.2f}"**.format(total\_cost))  
 print(**"Taxis are now:"**)  
 show\_cars(cars)  
  
  
def display\_taxis(taxis):  
 *"""Display numbered list of taxis."""* for i, taxi in enumerate(taxis):  
 print(**"{} - {}"**.format(i, taxi))  
  
  
main()