Oop

hot shocolate system

main

//import 'package:flutter/material.dart';  
import 'package:hotshproject/hotshoc.dart';  
  
void main() {  
 print('n');  
 //object  
 var object1 = hotshoclate();  
 // object1.makeorder(1);  
 object1.hotshlevel = 100;  
 object1.waterLevel = 500;  
 object1.makeorder(1);  
 print(object1.hotshlevel);  
}

Hotshoc class

class hotshoclate{  
 List hotshsize=[7,9,8];  
 double waterLevel=1000;  
 double hotshlevel=4000;  
  
  
 void turnonoption(){  
 print('option chosn');  
 }  
 void turnonoptioff(){  
 print('option close');  
 }  
  
 bool iswaterenough(int hotshsize ){  
 if(hotshsize==1&& waterLevel==500)  
 {  
 return true;  
 }  
 else{  
 return false;}  
 }  
 bool ishotenough(int hotshsize){  
 if(hotshsize==1&& hotshlevel==10)  
 {  
 return true;  
 }  
 else{  
 return false;}  
 //return true;  
 }  
 void warnhotshoclevellow(){  
  
 }  
  
 void makeorder(int hotshsize)  
 {  
 turnonoptioff();  
 if(hotshsize==1){  
 bool waterenough =iswaterenough( hotshsize);  
 bool hotenouh=ishotenough( hotshsize);  
 if (waterenough&&hotenouh)  
 {  
 //after make decress the water and leave the option button on  
 waterLevel -=500;  
 hotshlevel -=10;  
 print('hoschready');  
 turnonoption();  
 }  
 else{  
 print('not enough');  
 }  
  
 }  
 }  
}

……………………..

Constructor

in class

hotshoclate({List ?l,double ? h,double ?s}){  
 this.hotshsize=l!;  
 this.waterLevel=h!;  
 this.hotshlevel=s!;  
}

in main

import 'package:hotshproject/hotshoc.dart';  
  
void main() {  
 print('n');  
 //object  
 List hotshsize=[1];  
 //var object1 = hotshoclate( hotshsize,100,500);  
 var object=hotshoclate(l:hotshsize,h:100,s: 500);  
 // object1.makeorder(1);  
 //object1.hotshlevel = 100;  
 //object1.waterLevel = 500;  
 //object1.makeorder(1);  
 //print(object1.hotshlevel);  
 print(object.hotshlevel);  
  
}

………………..

Encapsulation

Put \_before variable then use set, get to get it

……………………………………………………………………………

train system

main

void main() {  
 //runApp(const MyApp());  
  
 final List<Seat> b = [  
 Seat(type: "yo", price: "50pound"),  
 Seat(type: "ra", price: "70pound"),  
 ];  
//call methods or objects  
 RaTrain n1 = RaTrain(id: "123", seats: b);  
  
 //call polymorphsm refre object  
 Train nn = RaTrain(id: "12", seats: b);  
 YoTrain nnn = YoTrain(id: "123", seats: b);  
  
 nn.bookEconomy();  
 print(n1.seats);  
 print(n1.createBookMessage());  
  
 //nnn.id;  
 //print("Trin\_id ${nnn.id} ");  
 //n1.id = "234";  
 //print("Trin\_id ${n1.id} ");  
 //print(leetManager().addTrain(nn));  
}

train

import 'seat.dart';  
  
class Train {  
 String id;  
 //list not arry you can add with any types generic  
 List<Seat> seats;  
 //constructor  
 Train({  
 required this.id,  
 required this.seats,  
 });  
  
 void bookEconomy() {}  
 void bookbusinus() {}  
}

Retrain

import 'package:train/classes/seat.dart';  
//import 'dart:ffi';  
import 'train.dart';  
  
//extendes inhertance  
class RaTrain extends Train {  
 //add function not in parent or attribtes  
 List<String> services = List.empty();  
  
 RaTrain({required String id, required List<Seat> seats})  
 : super(id: id, seats: seats);  
  
 //polymorph>ovride method  
  
 @override  
 void bookEconomy() {  
 print("book from rratrain");  
 }  
  
 Future<String> fetchUserBook() =>  
 // Imagine that this function is more complex and slow.  
 Future.delayed(  
 const Duration(seconds: 1),  
 () => 'Book done now',  
 );  
  
 String createBookMessage() {  
 var order = fetchUserBook();  
 return 'Your order is: $order';  
 }  
}

yo train

import 'package:train/classes/seat.dart';  
import 'package:train/classes/train.dart';  
  
class YoTrain extends Train {  
 YoTrain({required String id, required List<Seat> seats})  
 : super(id: id, seats: seats);  
}

seat

// ignore\_for\_file: public\_member\_api\_docs, sort\_constructors\_first  
class Seat {  
 String type;  
 String price;  
 Seat({  
 required this.type,  
 required this.price,  
 });  
}