Provider

First we will add the provider in the root level so that we can access it in the application from top to bottom or from to root to other part that's why we will add it into main.dart file.

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

import './screens/cart_screens.dart';
import './screens/product_detail_screen.dart';
import './screens/products_overview_screens.dart';
import './providers/products.dart';
import './providers/cart.dart';
import './providers/orders.dart';
import './screens/orders_screen.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (ctx) => Products(),
        ),
        ChangeNotifierProvider(
          create: (ctx) => Cart(),
        ),
        ChangeNotifierProvider(
          create: (ctx) => Orders(),
        ),
      ],
      child: MaterialApp(
        title: 'MyShop',
        theme: ThemeData(
          primarySwatch: Colors.purple,
          accentColor: Colors.deepOrange,
          fontFamily: 'Lato',
        ),
        home: ProductsOverviewScreen(),
        routes: {
          ProductDetailScreen.routeName: (ctx) =>
ProductDetailScreen(),
          CartScreen.routeName: (ctx) => CartScreen(),
          OrdersScreen.routeName: (ctx) => OrdersScreen(),
        },
      ),
    );
  }
}
```

Here I am Using more than one Provider that’s why I add it in tha main.dart file as a MultiProvider.

We will add a folder of Providers in our Application

In this folder I added a filer cart.dart

In this file we can add different methods and use this into our widgets and Screens.

```
import 'package:flutter/foundation.dart';

class CartItem {
  final String id;
  final String title;
  final int quantity;
  final double price;

  CartItem({
    @required this.id,
    @required this.title,
    @required this.quantity,
    @required this.price,
  });
}

class Cart with ChangeNotifier {
  Map<String, CartItem> _items = {};
```

```dart
Map<String, CartItem> get items {
  return {..._items};
}

int get itemCount {
  return _items.length;
}

double get totalAmount {
  var total = 0.0;
  _items.forEach((key, cartItem) {
    total += cartItem.price * cartItem.quantity;
  });
  return total;
}

void addItem(String productId, double price, String title) {
  if (_items.containsKey(productId)) {
    //change quantity
    _items.update(
        productId,
        (existingCartItem) => CartItem(
              id: existingCartItem.id,
              title: existingCartItem.title,
              quantity: existingCartItem.quantity + 1,
              price: existingCartItem.price,
            ));
  } else {
    _items.putIfAbsent(
        productId,
        () => CartItem(
              id: DateTime.now().toString(),
              title: title,
              quantity: 1,
              price: price,
            ));
  }
  notifyListeners();
}
```

```dart
  void removeItem(String productId) {
    _items.remove(productId);
    notifyListeners();
  }

  void removeSingleItem(String productId) {
    if (!_items.containsKey(productId)) {
      return;
    }
    if (_items[productId].quantity > 1) {
      _items.update(
          productId,
          (existingItem) => CartItem(
                id: existingItem.id,
                title: existingItem.title,
                quantity: existingItem.quantity - 1,
                price: existingItem.price,
              ));
    }else{
      _items.remove(productId);
    }
    notifyListeners();
  }

  void clear() {
    _items = {};
    notifyListeners();
  }
}
```

we can Use it into our Screens like this

first we can save the data into variable cart then we will access it using this variable.

```dart
final cart = Provider.of<Cart>(context);
```

for Example I am using the function of removeStringItem() into another Screen so I will access it like this cart. removeStringItem()

```
SnackBarAction(onPressed:(){
  cart.removeSingleItem(product.id);
} ,label: 'UNDO',),
```

And we can also Use Consumer to perform changing in a specific portion

So that we will only listen that portion rather than full widget

Consumer will not listen the child here

```
Consumer<Cart>(
  builder: (_, cart, ch) => Badge(
    child: ch,
    value: cart.itemCount.toString(),
  ),
  child: IconButton(
    icon: Icon(Icons.shopping_cart),
    onPressed: () {
      Navigator.of(context).pushNamed(CartScreen.routeName);
    },
  ),
),
```