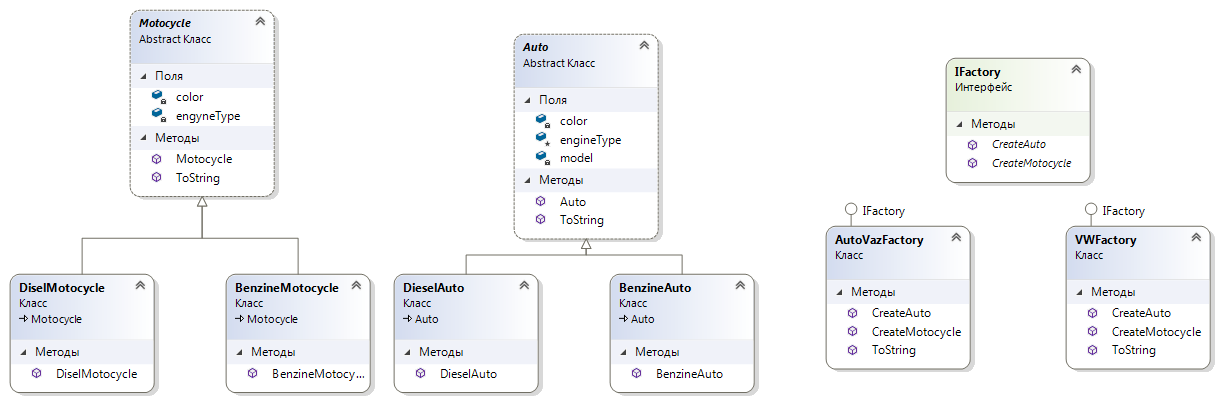
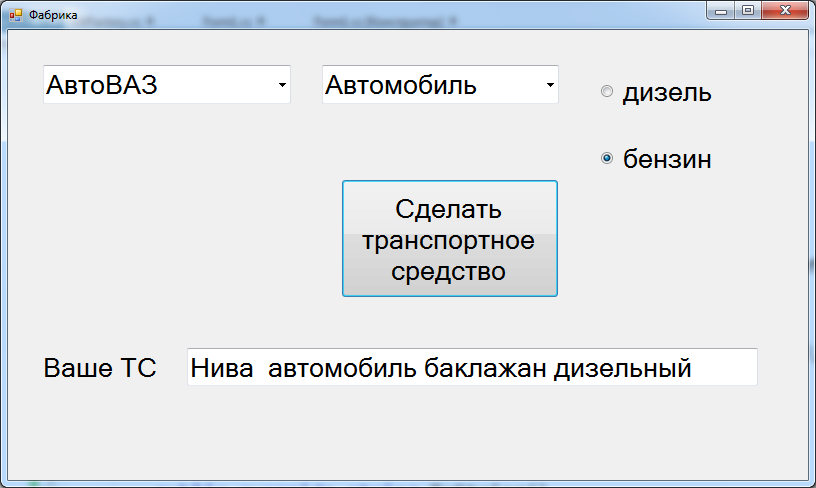
**Задание 1.**

Разработать систему классов, реализующую паттерн ***Абстрактная фабрика***

Классы-автомобильные заводы (фабрики «Автоваз» и «Фольксваген») должны создавать объекты двух видов: автомобили и мотоциклы (могут быть бензиновыми и дизельными).



В классе-клиенте пользователю предоставляется возможность выбрать фабрику и заказать транспортное средство. После чего ему выдается описание его заказа.



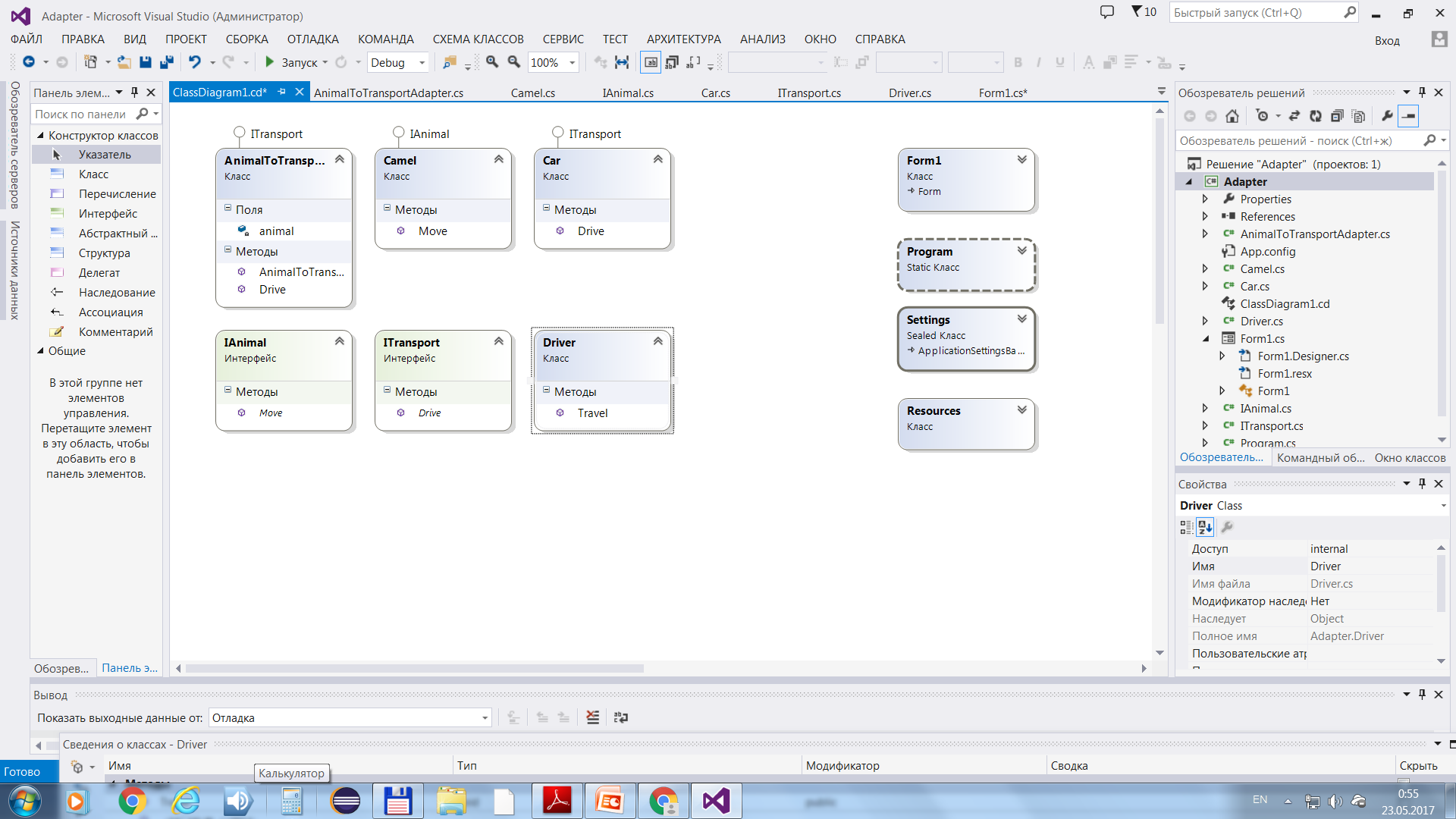
**Задание 2.**

Путешественник обычно путешествует на машине. Но в какой-то момент ему приходится передвигаться по пескам пустыни, или перемещаться по горной местности, или перелетать через океан (тогда он не может ехать на машине). Зато он может использовать для передвижения верблюда, лошадь или слона.

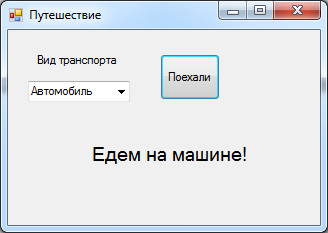
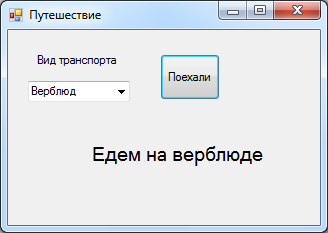
Разработать систему классов, реализующую паттерн ***Адаптер***

Класс **Автомобиль** представляет объекты, которые используются клиентом (Target).

Класс **Верблюд** представляет адаптируемый класс (Adaptee).

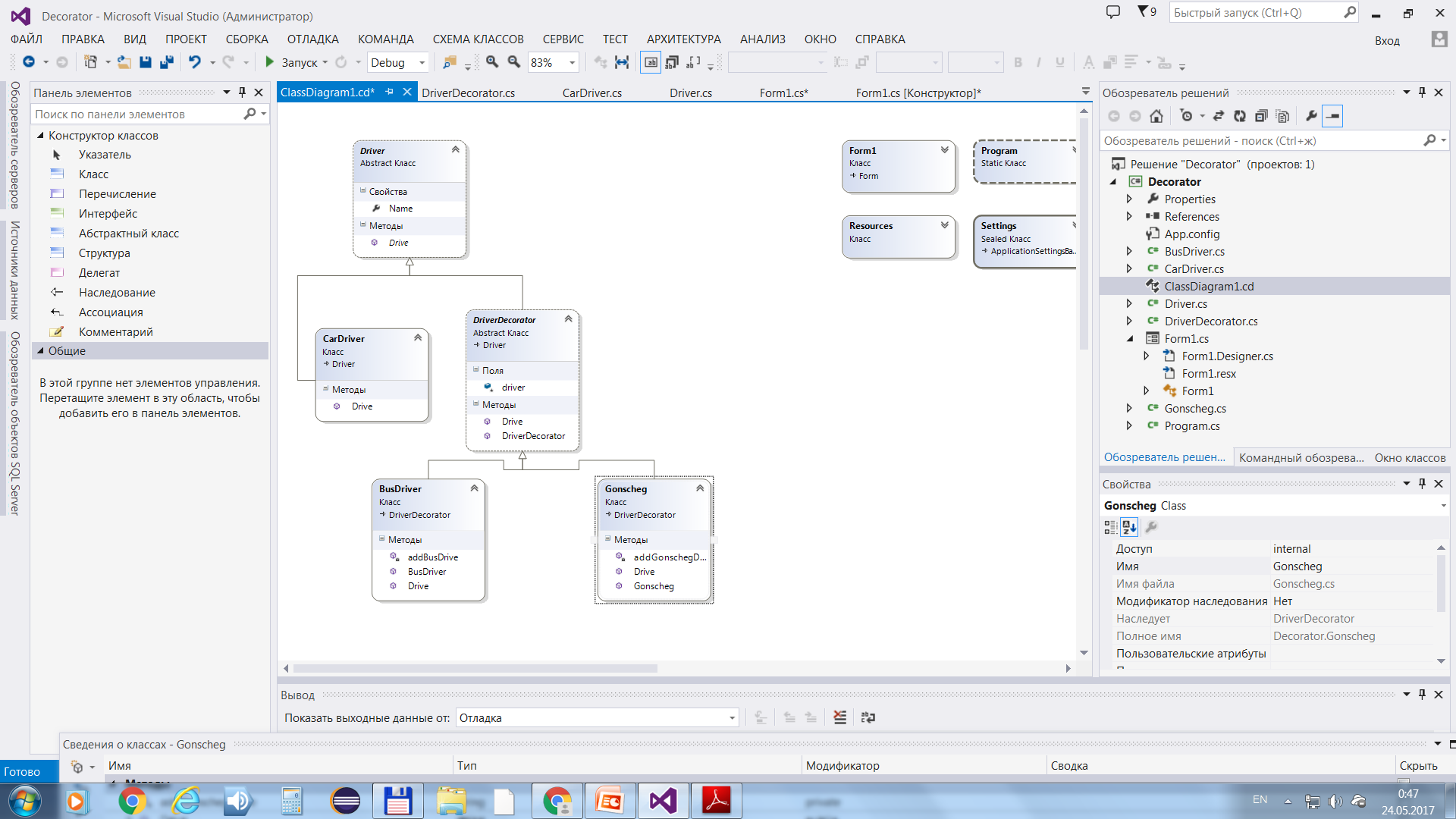


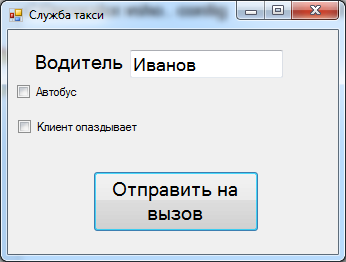
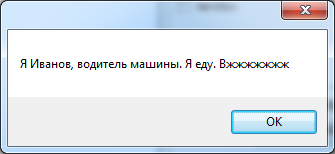
В классе-клиенте пользователю предоставляется возможность выбрать средство передвижения.

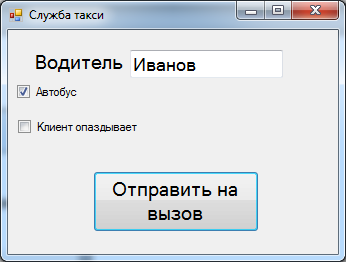
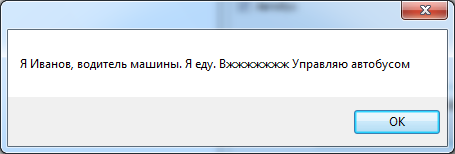
 

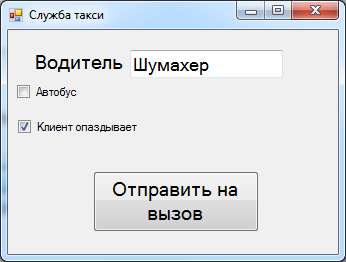
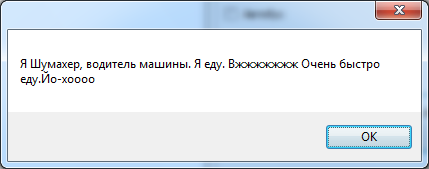
**Задание 3.**

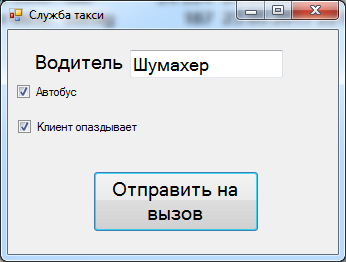
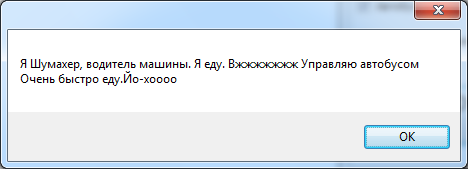
Разработать систему классов для приложения службы заказа такси, реализующую паттерн ***Декоратор (Decorator)***



**Задание 4.**

Разработать калькулятор для расчета цены заказа.

Для описания товара в корзине покупателя создать класс *ProductInCart*.

class ProductInCart

{

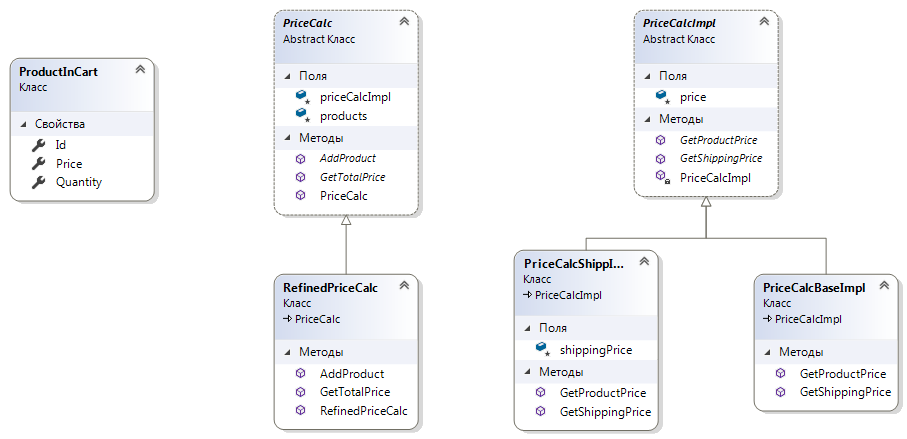
public uint Id { get; set; }

public uint Quantity { get; set; }

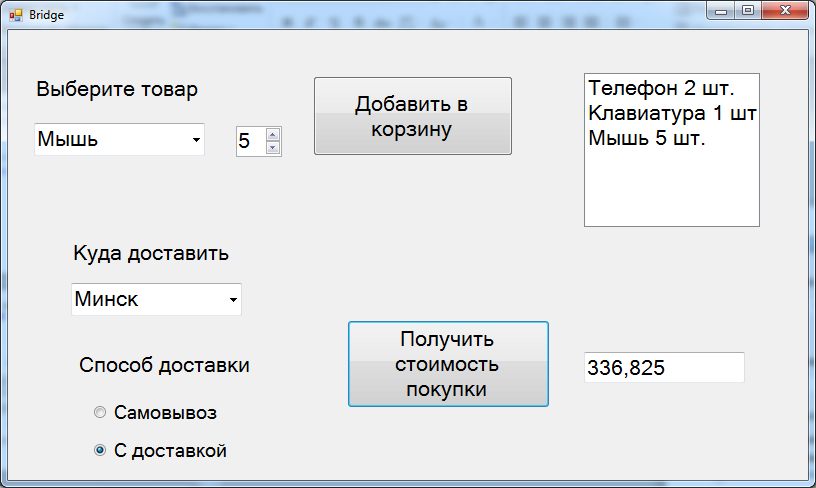
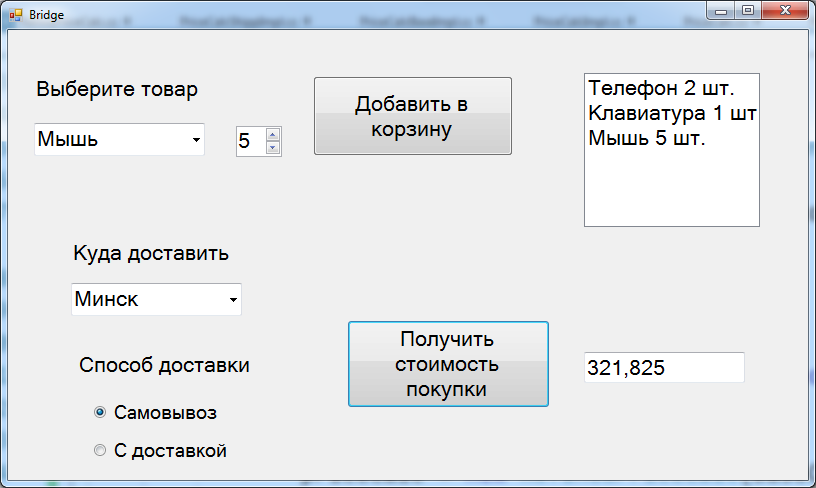
public decimal Price { get; set; }

}

При построении системы классов использовать паттерн **Мост.**

Интерфейс абстракции должен удовлетворять потребности клиента: добавлять позиции товаров и получать стоимость всего заказа.

В реализацию включить основные функции калькулятора: запросы цены единицы товара и стоимости доставки.



Пример реализации метода AddProduct:

public override void AddProduct(uint itemId, uint itemQuantity)

{

decimal price=priceCalcImpl.GetProductPrice(itemId);

products.Add(new ProductInCart { Id = itemId, Price = price, Quantity = itemQuantity });

}

Пример реализации метода GetTotalPrice:

public override decimal GetTotalPrice(string shippingTo)

{

return (products.Sum(p=>p.Price)+priceCalcImpl.GetShippingPrice(shippingTo));

}