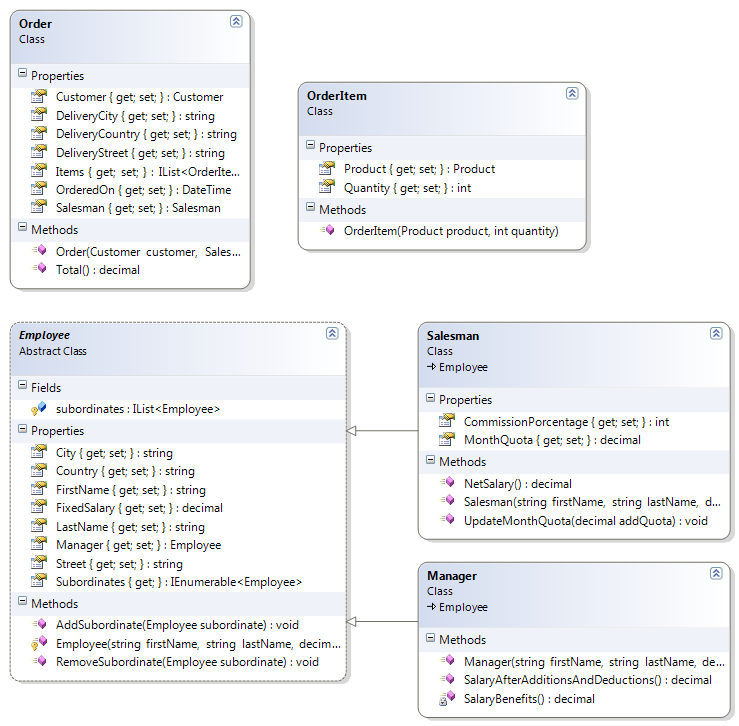
1. **FIRST COURSE**

**INITIAL TEE**

* **CLASS DIAGRAM**



* **CODE**

public class Order  
    {  
        public Customer Customer { get; private set; }  
   
        public Salesman Salesman { get; private set; }  
   
        public DateTime OrderedOn { get; private set; }  
   
        public string DeliveryStreet { get; private set; }  
   
        public string DeliveryCity { get; private set; }  
   
        public string DeliveryCountry { get; private set; }  
   
        public IList<OrderItem> Items { get; private set; }  
   
        public Order(Customer customer, Salesman salesman, string deliveryStreet, string deliveryCity, string deliveryCountry, DateTime orderedOn)  
        {  
            Customer = customer;  
            Salesman = salesman;  
            DeliveryStreet = deliveryStreet;  
            DeliveryCity = deliveryCity;  
            DeliveryCountry = deliveryCountry;  
            OrderedOn = orderedOn;  
            Items = new List<OrderItem>();  
        }  
   
        public decimal Total()  
        {  
            decimal totalAmount = 0;  
            foreach (var item in this.Items)  
            {  
                decimal totalItems = item.Product.UnitPrice \* item.Quantity;  
                if (item.Product.Category == ProductCategory.Accessories)  
                {  
                    decimal booksDiscount = 0;  
                    if (totalItems >= 100)  
                    {  
                        booksDiscount = totalItems \* 10 / 100;  
                    }  
                    totalItems = totalItems - booksDiscount;  
                }  
                if (item.Product.Category == ProductCategory.Bikes)  
                {  
                    totalItems = totalItems - totalItems \* 20 / 100;  
                }  
                if (item.Product.Category == ProductCategory.Cloathing)  
                {  
                    decimal cloathingDiscount = 0;  
                    if (item.Quantity > 2)  
                    {  
                        cloathingDiscount = item.Product.UnitPrice;  
                    }  
                    totalItems = totalItems - cloathingDiscount;  
                }  
                totalAmount += totalItems;  
            }  
   
            if (this.DeliveryCountry == "USA")  
            {  
                //totalAmount=totalItemAmount + tax + 0 shipping  
                return totalAmount + totalAmount \* 5 / 100;  
            }  
   
            //totalAmount=totalItemAmount + tax + 15 shipping  
            return totalAmount + totalAmount \* 5 / 100 + 15;  
        }  
    }

    public class OrderItem  
    {  
        public Product Product { get; private set; }  
   
        public int Quantity { get; private set; }  
   
        public OrderItem(Product product, int quantity)  
        {  
            Product = product;  
            Quantity = quantity;  
        }  
    }

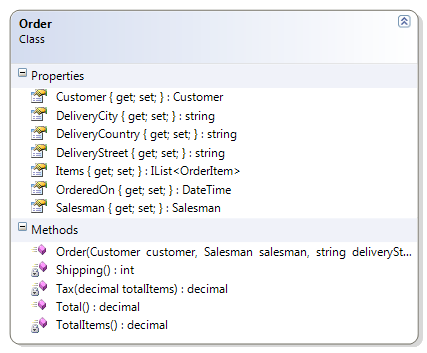
    public abstract class Employee  
    {  
        public string FirstName { get; private set; }  
   
        public string LastName { get; private set; }  
   
        public decimal FixedSalary { get; private set; }  
   
        public Employee Manager { get; internal set; }  
   
        public string Street { get; set; }  
   
        public string City { get; set; }  
   
        public string Country { get; set; }  
   
        protected IList<Employee> subordinates = new List<Employee>();  
   
        public IEnumerable<Employee> Subordinates  
        {  
            get { return subordinates.ToArray(); }  
        }  
   
        protected Employee(string firstName, string lastName, decimal fixedSalary)  
        {  
            this.FirstName = firstName;  
            this.LastName = lastName;  
            this.FixedSalary = fixedSalary;  
        }  
   
        public void AddSubordinate(Employee subordinate)  
        {  
            subordinates.Add(subordinate);  
            subordinate.Manager = this;  
        }  
   
        public void RemoveSubordinate(Employee subordinate)  
        {  
            subordinates.Remove(subordinate);  
            subordinate.Manager = null;  
        }  
    }

    public class Salesman : Employee  
    {  
        public int CommissionPorcentage { get; set; }  
   
        public decimal MonthQuota { get; private set; }  
   
        public Salesman(string firstName, string lastName, decimal fixedSalary, int commissionPorcentage)  
            : base(firstName, lastName, fixedSalary)  
        {  
            this.CommissionPorcentage = commissionPorcentage;  
        }  
   
        public decimal NetSalary()  
        {  
            decimal benefits = this.MonthQuota \* this.CommissionPorcentage / 100;  
            decimal pensionFounds = this.FixedSalary \* 10 / 100;  
            decimal tax = 0;  
            if (FixedSalary > 3500)  
                tax = FixedSalary \* 5 / 100;  
            return this.FixedSalary + benefits - pensionFounds - tax;  
        }  
   
        public void UpdateMonthQuota(decimal addQuota)  
        {  
            MonthQuota = MonthQuota + addQuota;  
        }  
    }

        public class Manager : Employee  
    {  
        public Manager(string firstName, string lastName, decimal fixedSalary)  
            : base(firstName, lastName, fixedSalary)  
        {  
        }  
          
        public decimal SalaryAfterAdditionsAndDeductions()  
        {  
            decimal benefits = SalaryBenefits();  
            decimal pensionFounds = this.FixedSalary \* 10 / 100;  
            decimal tax = 0;  
            if (FixedSalary > 3500)  
                tax = FixedSalary \* 5 / 100;  
            return this.FixedSalary + benefits - pensionFounds - tax;  
        }  
   
        private decimal SalaryBenefits()  
        {  
            return this.subordinates.Count \* 20;  
        }  
    }

**FIRST HOLE**

* **CLASS DIAGRAM**



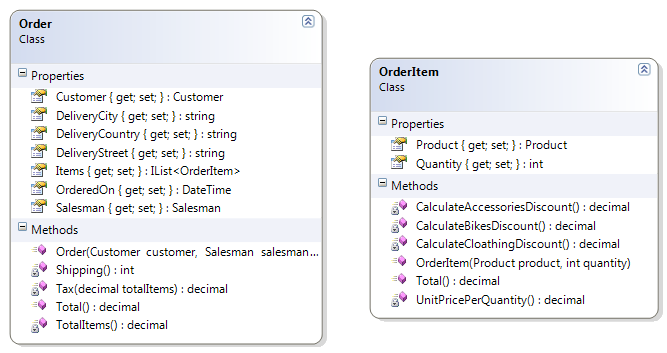
* **CODE**

    public class Order  
    {  
        public Customer Customer { get; private set; }  
   
        public Salesman Salesman { get; private set; }  
   
        public DateTime OrderedOn { get; private set; }  
   
        public string DeliveryStreet { get; private set; }  
   
        public string DeliveryCity { get; private set; }  
   
        public string DeliveryCountry { get; private set; }  
   
        public IList<OrderItem> Items { get; private set; }  
   
        public Order(Customer customer, Salesman salesman, string deliveryStreet, string deliveryCity, string deliveryCountry, DateTime orderedOn)  
        {  
            Customer = customer;  
            Salesman = salesman;  
            DeliveryStreet = deliveryStreet;  
            DeliveryCity = deliveryCity;  
            DeliveryCountry = deliveryCountry;  
            OrderedOn = orderedOn;  
            Items = new List<OrderItem>();  
        }

**public decimal Total()  
        {  
            var totalItems = this.TotalItems();  
            var tax = this.Tax(totalItems);  
            var shipping = this.Shipping();  
   
            return totalItems + tax + shipping;  
        }  
   
        private int Shipping()  
        {  
            int shipping = 15;  
            if (this.DeliveryCountry == "USA")  
            {  
                shipping = 0;  
            }  
            return shipping;  
        }  
   
        private decimal Tax(decimal totalItems)  
        {  
            return totalItems \* 5 / 100;  
        }  
   
        private decimal TotalItems()  
        {  
            decimal totalItems = 0;  
            foreach (var item in this.Items)  
            {  
                decimal itemAmount = item.Product.UnitPrice \* item.Quantity;  
                decimal discount = 0;  
                if (item.Product.Category == ProductCategory.Accessories)  
                {  
                    discount = 0;  
                    if (itemAmount >= 100)  
                    {  
                        discount = itemAmount \* 10 / 100;  
                    }  
                }  
                if (item.Product.Category == ProductCategory.Bikes)  
                {  
                    discount = itemAmount \* 20 / 100;  
                }  
                if (item.Product.Category == ProductCategory.Cloathing)  
                {  
                    discount = 0;  
                    if (item.Quantity > 2)  
                    {  
                        discount = item.Product.UnitPrice;  
                    }  
                }  
                itemAmount = itemAmount - discount;  
                totalItems += itemAmount;  
            }  
            return totalItems;  
        }**  
    }

**SECOND HOLE**

* **CLASS DIAGRAM**



* **CODE**

    public class Order  
    {  
        public Customer Customer { get; private set; }  
   
        public Salesman Salesman { get; private set; }  
   
        public DateTime OrderedOn { get; private set; }  
   
        public string DeliveryStreet { get; private set; }  
   
        public string DeliveryCity { get; private set; }  
   
        public string DeliveryCountry { get; private set; }  
   
        public IList<OrderItem> Items { get; private set; }  
   
        public Order(Customer customer, Salesman salesman, string deliveryStreet, string deliveryCity, string deliveryCountry, DateTime orderedOn)  
        {  
            Customer = customer;  
            Salesman = salesman;  
            DeliveryStreet = deliveryStreet;  
            DeliveryCity = deliveryCity;  
            DeliveryCountry = deliveryCountry;  
            OrderedOn = orderedOn;  
            Items = new List<OrderItem>();  
        }

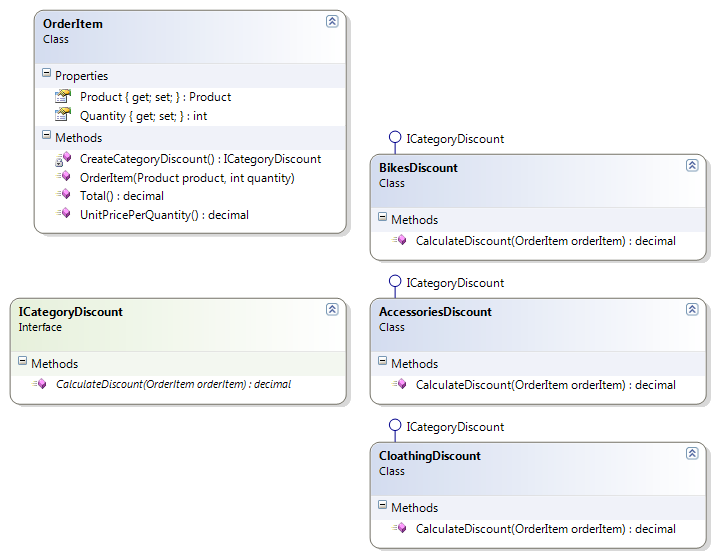
        public decimal Total()  
        {  
            var totalItems = this.TotalItems();  
            var tax = this.Tax(totalItems);  
            var shipping = this.Shipping();  
   
            return totalItems + tax + shipping;  
        }  
   
        private int Shipping()  
        {  
            int shipping = 15;  
            if (this.DeliveryCountry == "USA")  
            {  
                shipping = 0;  
            }  
            return shipping;  
        }  
   
        private decimal Tax(decimal totalItems)  
        {  
            var tax = totalItems \* 5 / 100;  
            return tax;  
        }  
   
**private decimal TotalItems()  
        {  
            decimal totalItems = 0;  
            foreach (var item in this.Items)  
            {  
                totalItems += item.Total();  
            }  
            return totalItems;  
        }**  
    }

    public class OrderItem  
    {  
        public Product Product { get; private set; }  
   
        public int Quantity { get; private set; }  
   
        public OrderItem(Product product, int quantity)  
        {  
            Product = product;  
            Quantity = quantity;  
        }

        **public decimal Total()  
        {  
            decimal discount = 0;  
            if (Product.Category == ProductCategory.Accessories)  
            {  
                discount = this.CalculateAccessoriesDiscount();  
            }  
            if (Product.Category == ProductCategory.Bikes)  
            {  
                discount = this.CalculateBikesDiscount();  
            }  
            if (Product.Category == ProductCategory.Cloathing)  
            {  
                discount = this.CalculateCloathingDiscount();  
            }  
            return this.UnitPricePerQuantity() - discount;  
        }  
   
        private decimal CalculateAccessoriesDiscount()  
        {  
            decimal discount = 0;  
            if (this.UnitPricePerQuantity() >= 100)  
            {  
                discount = this.UnitPricePerQuantity() \* 10 / 100;  
            }  
            return discount;  
        }  
   
        private decimal CalculateBikesDiscount()  
        {  
            return this.UnitPricePerQuantity() \* 20 / 100;  
        }  
   
        private decimal CalculateCloathingDiscount()  
        {  
            decimal discount = 0;  
            if (this.Quantity > 2)  
            {  
                discount = this.Product.UnitPrice;  
            }  
            return discount;  
        }  
   
        private decimal UnitPricePerQuantity()  
        {  
            return this.Product.UnitPrice \* this.Quantity;  
        }**    }

**THIRD HOLE**

* **CLASS DIAGRAM**



* **CODE**

    public class OrderItem  
    {  
        public Product Product { get; private set; }  
   
        public int Quantity { get; private set; }  
   
        public OrderItem(Product product, int quantity)  
        {  
            Product = product;  
            Quantity = quantity;  
        }  
   
**public decimal Total()  
        {  
            return this.UnitPricePerQuantity() - this.CreateCategoryDiscount().CalculateDiscount(this);  
        }**

**private ICategoryDiscount CreateCategoryDiscount()  
        {  
            ICategoryDiscount categoryDiscount = null;  
            if (this.Product.Category == ProductCategory.Accessories)  
            {  
                categoryDiscount = new AccessoriesDiscount();  
            }  
            if (this.Product.Category == ProductCategory.Bikes)  
            {  
                categoryDiscount = new BikesDiscount();  
            }  
            if (this.Product.Category == ProductCategory.Cloathing)  
            {  
                categoryDiscount = new CloathingDiscount();  
            }  
            return categoryDiscount;  
        }**   
        public decimal UnitPricePerQuantity()  
        {  
            return this.Product.UnitPrice \* this.Quantity;  
        }  
    }

**public interface ICategoryDiscount{  
        decimal CalculateDiscount(OrderItem orderItem);  
    }**

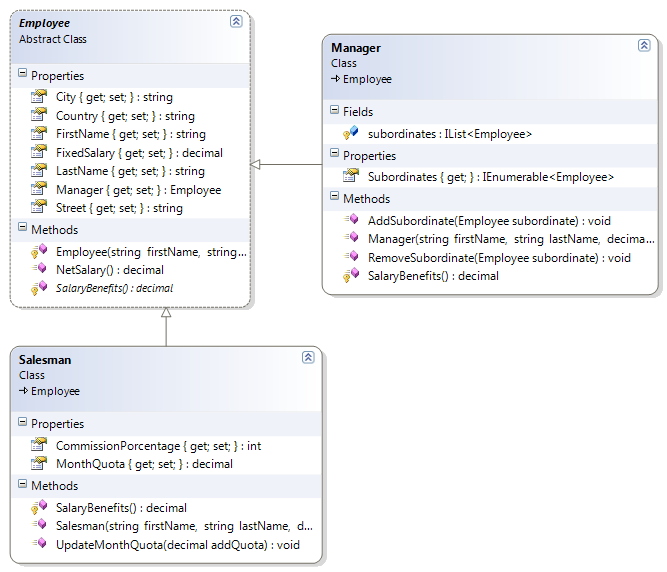
**public class AccessoriesDiscount : ICategoryDiscount{  
        public decimal CalculateDiscount(OrderItem orderItem)  
        {  
            decimal discount = 0;  
            if (orderItem.UnitPricePerQuantity() >= 100)}{  
                discount = orderItem.UnitPricePerQuantity() \* 10 / 100;  
            }  
            return discount;  
        }  
    }**

**public class BikesDiscount : ICategoryDiscount  
        public decimal CalculateDiscount(OrderItem orderItem)  
        {  
            return orderItem.UnitPricePerQuantity() \* 20 / 100;  
        }  
    }**

**public class CloathingDiscount : ICategoryDiscount{  
        public decimal CalculateDiscount(OrderItem orderItem)  
        {  
            decimal discount = 0;  
            if (orderItem.Quantity > 2)  
            {  
                discount = orderItem.Product.UnitPrice;  
            }  
            return discount;  
        }  
    }**

**FOURTH HOLE**

* **CLASS DIAGRAM**



* **CODE**

    public abstract class Employee  
    {  
        public string FirstName { get; private set; }  
   
        public string LastName { get; private set; }  
   
        public decimal FixedSalary { get; private set; }  
   
        public Employee Manager { get; internal set; }  
   
        public string Street { get; set; }  
   
        public string City { get; set; }  
   
        public string Country { get; set; }  
   
        protected Employee(string firstName, string lastName, decimal fixedSalary)  
        {  
            this.FirstName = firstName;  
            this.LastName = lastName;  
            this.FixedSalary = fixedSalary;  
        }  
   
**public decimal NetSalary()  
        {  
            decimal benefits = SalaryBenefits();  
            decimal pensionFounds = this.FixedSalary \* 10 / 100;  
            decimal tax = 0;  
            if (this.FixedSalary > 3500)  
                tax = this.FixedSalary \* 5 / 100;  
            return this.FixedSalary + benefits - pensionFounds - tax;  
        }  
   
        protected abstract decimal SalaryBenefits();**  
    }

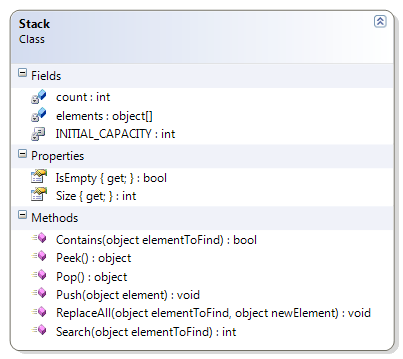
public class Salesman : Employee  
    {  
        public int CommissionPorcentage{ get; set; }  
   
        public decimal MonthQuota{ get; private set; }  
   
        public Salesman(string firstName, string lastName, decimal fixedSalary, int commissionPorcentage): base(firstName, lastName, fixedSalary) {  
            this.CommissionPorcentage = commissionPorcentage;  
        }  
   
**protected override decimal SalaryBenefits()  
        {  
            return this.MonthQuota \* this.CommissionPorcentage / 100;  
        }**  
   
        public void UpdateMonthQuota(decimal addQuota)  
        {  
            MonthQuota = MonthQuota + addQuota;  
        }  
    }

public class Manager : Employee  
    {  
        protected IList<Employee> subordinates = new List<Employee>();  
   
        public Manager(string firstName, string lastName, decimal fixedSalary)  
            : base(firstName, lastName, fixedSalary){        }  
   
**protected override decimal SalaryBenefits()  
        {  
            return this.subordinates.Count \* 20;  
        }  
   
        public IEnumerable<Employee> Subordinates  
        {  
            get { return subordinates.ToArray(); }  
        }  
   
        public void AddSubordinate(Employee subordinate)  
        {  
            subordinates.Add(subordinate);  
            subordinate.Manager = this;  
        }  
   
        public void RemoveSubordinate(Employee subordinate)  
        {  
            subordinates.Remove(subordinate);  
            subordinate.Manager = null;  
        }**  
    }

1. **SECOND COURSE**

**TEE**

* **CLASS DIAGRAM**

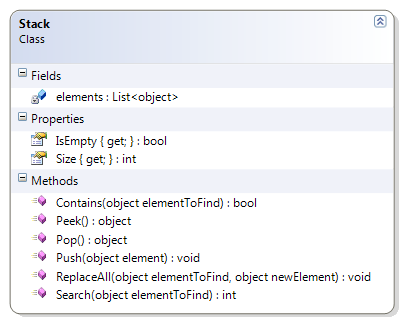


* **CODE**

    public class Stack  
    {  
        private const int INITIAL\_CAPACITY = 5;  
        private object[] elements = new object[INITIAL\_CAPACITY];  
        private int count;  
   
        public bool IsEmpty  
        {  
            get { return count == 0; }  
        }  
   
        public int Size  
        {  
            get { return count; }  
        }  
   
        public void Push(object element)  
        {  
            if (count + 1 > this.elements.Length)  
            {  
                object[] temp = new object[2 \* this.elements.Length];  
                Array.Copy(elements, temp, count);  
                elements = temp;  
            }  
            elements[count] = element;  
            count++;  
        }  
   
        public object Pop()  
        {  
            if (IsEmpty)  
                throw new InvalidOperationException();  
            object element = elements[count - 1];  
            count--;  
            return element;  
        }  
   
        public object Peek()  
        {  
            return elements[count - 1];  
        }  
   
        public bool Contains(object elementToFind)  
        {  
            for (int i = 0; i < count; i++)  
            {  
                if (elementToFind == elements[i])  
                {  
                    return true;  
                }  
            }  
            return false;  
        }  
   
        public int Search(object elementToFind)  
        {  
            for (int i = 1; i <= count; i++)  
            {  
                if (elementToFind == elements[count - i])  
                {  
                    return i;  
                }  
            }  
            return -1;  
        }  
   
        public void ReplaceAll(object elementToFind, object newElement)  
        {  
            for (int i = count - 1; i >= 0; i--)  
            {  
                if (elementToFind == elements[i])  
                {  
                    elements[i] = newElement;  
                }  
            }  
        }  
    }

**HOLE**

* **CLASS DIAGRAM**



* **CODE**

    public class Stack  
    {  
        private List<object> elements = new List<object>();  
   
        public bool IsEmpty  
        {  
            get { return Size == 0; }  
        }  
   
        public int Size  
        {  
            get { return elements.Count; }  
        }  
   
        public void Push(object element)  
        {  
            elements.Add(element);  
        }  
   
        public object Pop()  
        {  
            if (IsEmpty)  
                throw new InvalidOperationException();  
            object element = elements[Size - 1];  
            elements.RemoveAt(Size-1);  
            return element;  
        }

        public object Peek()  
        {  
            return elements[Size - 1];  
        }  
   
        public bool Contains(object elementToFind)  
        {  
            int indexOf = Search(elementToFind);  
            if (indexOf != -1)  
            {  
                return true;  
            }  
            return false;  
        }  
   
        public int Search(object elementToFind)  
        {  
            for (int i = 1; i <= Size; i++)  
            {  
                if (elementToFind == elements[Size-i])  
                {  
                    return i;  
                }  
            }  
            return -1;  
        }  
   
        public void ReplaceAll(object elementToFind, object newElement)  
        {  
            for (int i = Size - 1; i >= 0; i--)  
            {  
                if (elementToFind == elements[i])  
                {  
                    elements[i] = newElement;  
                }  
            }  
        }  
    }