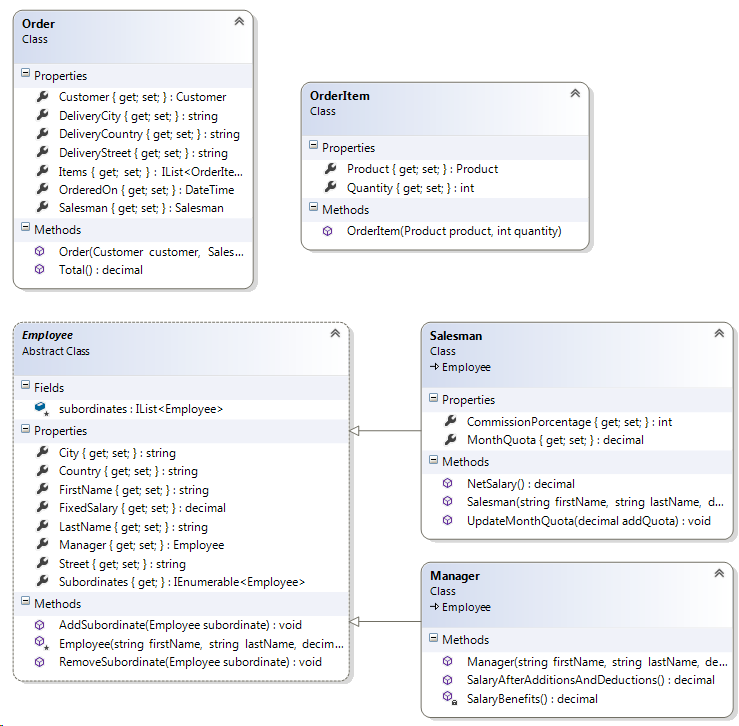
**FIRST COURSE**

****

**FIRST HOLE**



public class Order

{

public decimal Total()

{

decimal totalItems = 0;

foreach (var item in this.Items)

{

decimal totalItem = 0;

decimal itemAmount = item.Product.UnitPrice \* item.Quantity;

if (item.Product.Category == ProductCategory.Accessories)

{

decimal booksDiscount = 0;

if (itemAmount >= 100)

{

booksDiscount = itemAmount \* 10 / 100;

}

totalItem = itemAmount - booksDiscount;

}

if (item.Product.Category == ProductCategory.Bikes)

{

// 20% discount for Bikes

totalItem = itemAmount - itemAmount \* 20 / 100;

}

if (item.Product.Category == ProductCategory.Cloathing)

{

decimal cloathingDiscount = 0;

if (item.Quantity > 2)

{

cloathingDiscount = item.Product.UnitPrice;

}

totalItem = itemAmount - cloathingDiscount;

}

totalItems += totalItem;

}

if (this.DeliveryCountry == "USA")

{

//total=totalItems + tax + 0 shipping

return totalItems + totalItems \* 5 / 100;

}

//total=totalItems + tax + 15 shipping

return totalItems + totalItems \* 5 / 100 + 15;

}

}

**FIRST HOLE**



public class Order

{

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)

{

var itemAmount = TotalItem(item);

totalItems += itemAmount;

}

return totalItems;

}

private decimal TotalItem(OrderItem item)

{

decimal totalItem = 0;

decimal itemAmount = item.Product.UnitPrice\*item.Quantity;

if (item.Product.Category == ProductCategory.Accessories)

{

decimal booksDiscount = 0;

if (itemAmount >= 100)

{

booksDiscount = itemAmount\*10/100;

}

totalItem = itemAmount - booksDiscount;

}

if (item.Product.Category == ProductCategory.Bikes)

{

// 20% discount for Bikes

totalItem = itemAmount - itemAmount \* 20 / 100;

}

if (item.Product.Category == ProductCategory.Cloathing)

{

decimal cloathingDiscount = 0;

if (item.Quantity > 2)

{

cloathingDiscount = item.Product.UnitPrice;

}

totalItem = itemAmount - cloathingDiscount;

}

return totalItem;

}

}

**SECOND HOLE**



public class Order

{

private decimal TotalItems()

{

decimal totalItems = 0;

foreach (var item in this.Items)

{

totalItems += item.Total();

}

return totalItems;

}

}

public class OrderItem

{

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.ItemAmount() - discount;

}

private decimal CalculateAccessoriesDiscount()

{

decimal discount = 0;

if (this.ItemAmount() >= 100)

{

discount = this.ItemAmount() \* 10 / 100;

}

return discount;

}

private decimal CalculateBikesDiscount()

{

return this.ItemAmount() \* 20 / 100;

}

private decimal CalculateCloathingDiscount()

{

decimal discount = 0;

if (this.Quantity > 2)

{

discount = this.Product.UnitPrice;

}

return discount;

}

private decimal ItemAmount()

{

return this.Product.UnitPrice \* this.Quantity;

}

}

**THIRD HOLE**



public class OrderItem

{

public decimal Total()

{

return this.ItemAmount() - 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 interface ICategoryDiscount

{

decimal CalculateDiscount(OrderItem orderItem);

}

public class AccessoriesDiscount : ICategoryDiscount

{

public decimal CalculateDiscount(OrderItem orderItem)

{

decimal discount = 0;

if (orderItem.ItemAmount() >= 100)

{

discount = orderItem.ItemAmount() \* 10 / 100;

}

return discount;

}

}

public class BikesDiscount : ICategoryDiscount

{

public decimal CalculateDiscount(OrderItem orderItem)

{

return orderItem.ItemAmount() \* 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**

* **PREVIOUS HOLE**



public abstract class Employee

{

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 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 class Manager : Employee

{

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;

}

}

* **CURRENT HOLE**



public abstract class Employee

{

public decimal NetSalary()

{

return this.FixedSalary + SalaryBenefits() - PensionFounds() - Tax();

}

private decimal Tax()

{

decimal tax = 0;

if (this.FixedSalary > 3500)

tax = this.FixedSalary\*5/100;

return tax;

}

private decimal PensionFounds()

{

return this.FixedSalary \* 10 / 100;

}

protected abstract decimal SalaryBenefits();

}

public class Salesman : Employee

{

protected override decimal SalaryBenefits()

{

return this.MonthQuota \* this.CommissionPorcentage / 100;

}

}

public class Manager : Employee

{

protected IList<Employee> subordinates = new List<Employee>();

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;

}

}