**FIRST COURSE**

**FIRST HOLE**

public class Order {

public float total() {

float totalItems = totalItems();

float tax = tax(totalItems);

int shipping = shipping();

return totalItems + tax + shipping;

}

private float tax(float totalAmount) {

return totalAmount \* 5 / 100;

}

private int shipping() {

int shipping = 15;

if (this.deliveryCountry == "USA") {

shipping = 0;

}

return shipping;

}

private float totalItems() {

float totalAmount = 0;

for (OrderItem item : items) {

float discount=0;

float itemAmount = item.getProduct().getUnitPrice() \* item.getQuantity();

if (item.getProduct().getCategory() == ProductCategory.Accessories) {

discount = 0;

if (itemAmount >= 100) {

discount = itemAmount \* 10 / 100;

}

}

if (item.getProduct().getCategory() == ProductCategory.Components) {

discount = itemAmount \* 5 / 100;

}

if (item.getProduct().getCategory() == ProductCategory.Bikes) {

discount = itemAmount \* 20 / 100;

}

if (item.getProduct().getCategory() == ProductCategory.Cloathing) {

discount = 0;

if (item.getQuantity() > 2) {

discount = item.getProduct().getUnitPrice();

}

}

itemAmount = itemAmount - discount;

totalAmount += itemAmount;

}

return totalAmount;

}

}

**SECOND HOLE**

public class Order {

private float totalItems() {

float totalAmount = 0;

for (OrderItem item : items) {

totalAmount += item.total();

}

return totalAmount;

}

}

public class OrderItem {

float total() {

float discount=0;

if (getProduct().getCategory() == ProductCategory.Accessories) {

discount = calculateAccessoriesDiscount();

}

if (getProduct().getCategory() == ProductCategory.Components) {

discount = calculateComponentsDiscount();

}

if (getProduct().getCategory() == ProductCategory.Bikes) {

discount = calculateBikesDiscount();

}

if (getProduct().getCategory() == ProductCategory.Cloathing) {

discount = calculateCloathingDiscount();

}

return unitPricePerQuantity() - discount;

}

private float calculateAccessoriesDiscount() {

float discount = 0;

float unitPricePerQuantity = unitPricePerQuantity();

if (unitPricePerQuantity >= 100) {

discount = unitPricePerQuantity \* 10 / 100;

}

return discount;

}

private float calculateBikesDiscount() {

return unitPricePerQuantity() \* 20 / 100;

}

private float calculateCloathingDiscount() {

float discount = 0;

if (getQuantity() > 2) {

discount = getProduct().getUnitPrice();

}

return discount;

}

private float calculateComponentsDiscount() {

return unitPricePerQuantity() \* 5 / 100;

}

private float unitPricePerQuantity() {

return getProduct().getUnitPrice() \* getQuantity();

}

}

**THIRD HOLE**

public class OrderItem {

float total() {

return unitPricePerQuantity() - createCategoryDiscount().calculateDiscount(this);

}

private CategoryDiscount createCategoryDiscount() {

CategoryDiscount categoryDiscount=null;

if (getProduct().getCategory() == ProductCategory.Accessories) {

categoryDiscount = new AccessoriesDiscount();

}

if (getProduct().getCategory() == ProductCategory.Components) {

categoryDiscount = new ComponentsDiscount();

}

if (getProduct().getCategory() == ProductCategory.Bikes) {

categoryDiscount = new BikesDiscount();

}

if (getProduct().getCategory() == ProductCategory.Cloathing) {

categoryDiscount = new CloathingDiscount();

}

return categoryDiscount;

}

}

public interface CategoryDiscount {

float calculateDiscount(OrderItem orderItem);

}

public class AccessoriesDiscount implements CategoryDiscount {

public float calculateDiscount(OrderItem orderItem) {

float discount = 0;

float unitPricePerQuantity = orderItem.unitPricePerQuantity();

if (unitPricePerQuantity >= 100) {

discount = unitPricePerQuantity \* 10 / 100;

}

return discount;

}

}

public class BikesDiscount implements CategoryDiscount{

public float calculateDiscount(OrderItem orderItem) {

return orderItem.unitPricePerQuantity() \* 20 / 100;

}

}

public class CloathingDiscount implements CategoryDiscount {

public float calculateDiscount(OrderItem orderItem) {

float discount = 0;

if (orderItem.getQuantity() > 2) {

discount = orderItem.getProduct().getUnitPrice();

}

return discount;

}

}

public class ComponentsDiscount implements CategoryDiscount {

public float calculateDiscount(OrderItem orderItem) {

return orderItem.unitPricePerQuantity() \* 5 / 100;

}

}

**SECOND COURSE**

public class Stack {

private List<Object> elements = new ArrayList<Object>();

public boolean isEmpty()

{

return size() == 0;

}

public int size()

{

return elements.size();

}

public void push(Object element)

{

elements.add(element);

}

public Object pop()

{

if (isEmpty())

throw new IllegalStateException();

Object element = elements.get(size() - 1);

elements.remove(size()-1);

return element;

}

public Object peek()

{

return elements.get(size() - 1);

}

public boolean 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.get(size()-i))

{

return i;

}

}

return -1;

}

public void replaceAll(Object elementToFind, Object newElement)

{

for (int i = size() - 1; i >= 0; i--)

{

if (elementToFind == elements.get(i))

{

elements.set(i,newElement);

}

}

}

}