public class Value-meal extends Product {

private ArrayList<MItem> items;

public Value-meal(ArrayList<MItem> items) {

this.items = items;

}

public double total() {

double sum = 0.0; // Initialize sum to accumulate item prices

for (MItem item : items) { // Iterate through each item in the ArrayList

sum += item.getPrice(); // Add the price of each item to sum

}

return sum; // Return the total price of the items

}

// Other methods not shown

}

public class VM extends Product {

/\* constructor and instance variables not shown \*/

public static double calcVMPrice(VM burger) {

double originalPrice = burger.total(); // Get the total price of items in the burger value-meal

double discountedPrice = originalPrice \* 0.80; // Apply 20% discount

double roundedPrice = roundToNearestDimeUp(discountedPrice); // Round up to nearest 10 cents

double finalPrice = roundedPrice + 0.09; // Increase by $0.09

return finalPrice; // Return the calculated discounted price

}

public static double roundToNearestDimeUp(double d) {

double dInDimes = 10 \* d;

if (dInDimes - (int) dInDimes == 0) {

return d;

}

return ((int) (dInDimes + 1)) / 10.0; // Correct rounding logic to always round up

}

/\* other methods not shown \*/

}

public class VM extends Product {

/\* constructor and instance variables not shown \*/

public static VM highestPricedVM(ArrayList<VM> burgers) {

if (burgers == null || burgers.isEmpty()) {

return null; // Return null if the input list is null or empty

}

VM highestPricedVM = burgers.get(0); // Start with the first VM as the highest priced

double highestPrice = calcVMPrice(highestPricedVM); // Calculate its price

for (int i = 1; i < burgers.size(); i++) { // Iterate through the list starting from the second element

VM currentVM = burgers.get(i);

double currentPrice = calcVMPrice(currentVM); // Calculate current VM's price

if (currentPrice > highestPrice) { // Compare current VM's price with the highest price found so far

highestPricedVM = currentVM; // Update highestPricedVM

highestPrice = currentPrice; // Update highestPrice

}

}

return highestPricedVM; // Return the VM with the highest price

}

/\* other methods not shown \*/

}