A)Package - ElectronicProducts

1. Abstract Class- ElectronicProduct

VARIABLES-.

1. productName- The name of a product available in the electronic store.
2. price- The price of the electronic product.
3. quantity- the total no. of items bought from the store.
4. brand- the brand of the respective electronic product.

METHODS

1)getProductName- a getter to retrieve the product name.

2)setProductName(String productName)- a setter to set method name with “productName”.

3) getBrandName- a getter for brand name.

4) setBrandName(String band)- a setter to set method band name with “band”.

5)getQuantity- a getter to get quantity of electronic products.

6)setQuantity- a setter to set method name with “quantity”.

7)calculateTotalCost- an abstract method which will be used in calculating cost of devices in every device class.

2) Public class -AirConditioner extends ElectronicProduct

VARIABLES-  
1) coolingCapacity- field to store cooling capacity of the air conditioner.

* Overloaded constructor to initialize an AirConditioner object with specified details including brand.
* \* Throws InvalidProductDetailsException if cooling capacity is not greater than 0.

\* @param productName The name of the air conditioner.

\* @param price The price of the air conditioner.

\* @param quantity The quantity of the air conditioner.

\* @param coolingCapacity The cooling capacity of the air conditioner.

\* @param brand The brand of the air conditioner.

\* @throws InvalidProductDetailsException if cooling capacity is not greater than 0.

* Overloaded constructor to initialize an AirConditioner object with specified details including brand
* Throws InvalidProductDetailsException if cooling capacity is not greater than 0

METHODS

* getCoolingCapacity- Getter method to retrieve the cooling capacity of the air conditioner.
* setCoolingCapacity(Double coolingCapacity)- Setter method to set the cooling capacity of the air conditioner.
* calculateTotalCost- Method to calculate the total cost of the air conditioner based on price and quantity.

3) Public class-Dishwasher extends ElectronicProduct

VARIABLES-

1. numberOfRacks- field to store the number of racks of the dishwasher.

* Constructor to initialize a Dishwasher object with specified details
* Throws InvalidProductDetailsException if the number of racks is not greater than 0

@param productName The name of the dishwasher.

\* @param price The price of the dishwasher.

\* @param quantity The quantity of the dishwasher.

\* @param numberOfRacks The number of racks of the dishwasher.

\* @throws InvalidProductDetailsException if the number of racks is not greater than 0.

METHODS-

* getNumberOfRacks- Getter method to retrieve the number of racks of the dishwasher.
* setNumberOfRacks(Integer numberOfRacks)- Setter method to set the number of racks of the dishwasher.
* calculateTotalCost()- Method to calculate the total cost of the dishwasher based on price and quantity.

4) Public class-Refrigerator extends ElectronicProduct

VARIABLES-

1. capacity-field to store the capacity of the refrigerator.

* Constructor to initialize a Refrigerator object with specified details.
* Throws InvalidProductDetailsException if capacity is not greater than 0.

\* @param productName The name of the refrigerator.

\* @param price The price of the refrigerator.

\* @param quantity The quantity of the refrigerator.

\* @param capacity The capacity of the refrigerator.

\* @throws InvalidProductDetailsException if capacity is not greater than 0.

METHODS-

* getCapacity()- Getter method to retrieve the capacity of the refrigerator.
* setCapacity(Double capacity)- Setter method to set the capacity of the refrigerator.
* calculateTotalCost()- Method to calculate the total cost of the refrigerator based on price and quantity.

5) Public class InvalidProductDetailsException extends Exception-

* Custom exception class for invalid product details

VARIABLES-

1. serialVersionUID- Serial version UID for serialization.

6) Public class-Television extends ElectronicProduct

VARIABLES-

1. resolution- field to store the resolution of the television.

* Constructor to initialize a Television object with specified details.
* Throws InvalidProductDetailsException if resolution is not valid.\* @param productName The name of the television.
* \* @param price The price of the television.
* \* @param quantity The quantity of the television.
* \* @param resolution The resolution of the television.
* \* @throws InvalidProductDetailsException if resolution is not valid.

METHODS-

* getResolution()- Getter method to retrieve the resolution of the television.
* setResolution(String resolution)-Setter method to set the resolution of the television.
* calculateTotalCost()- Method to calculate the total cost of the television based on price and quantity.

7) Public class-WashingMachine extends ElectronicProduct

VARIABLES-

1. capacity-Private field to store the capacity of the washing machine

* \* Constructor to initialize a WashingMachine object with specified details.
* \* Throws InvalidProductDetailsException if capacity is not greater than 0.
* \*
* \* @param productName The name of the washing machine.
* \* @param price The price of the washing machine.
* \* @param quantity The quantity of the washing machine.
* \* @param capacity The capacity of the washing machine.
* \* @throws InvalidProductDetailsException if capacity is not greater than 0.

METHODS-

* getCapacity()-Getter method to retrieve the capacity of the washing machine.
* setCapacity(Double capacity)- Setter method to set the capacity of the washing machine.
* calculateTotalCost()- Method to calculate the total cost of the washing machine based on price and quantity.

B) Package - ElectronicStore

1. Public class EmployeeManagement.

VARIABLES AND ARRAYS.

1. Employee[] employees-Array to store the employees.
2. employeeCount-Variable to keep track of the number of employees.

METHODS-

1. void hireEmployee(Employee employee)-Method to hire an employee.
2. void fireEmployee(Employee employee)-Method to fire an employee.
3. void displayEmployees()-Method to display the list of employees.

2) Public class ProductManagement-Class responsible for managing the inventory of electronic products

VARIABLES AND ARRAYS.

1. ElectronicProduct[] inventory- Array to store the inventory of electronic products.
2. inventorySize- Variable to keep track of the size of the inventory.

METHODS-

1. void addProduct(ElectronicProduct product)- Method to add a product to the inventory.
2. void addProduct(String name, String brand)- Overloaded method to add products to the inventory based on name and brand.
3. void removeProduct(ElectronicProduct product)- Method to remove a product from the inventory.
4. void removeProduct(ElectronicProduct... products)- Varargs overloaded method to remove multiple products from the inventory.
5. void displayInventory()-Method to display the inventory.
6. void purchaseProduct(String productName, int quantity, double purchasePrice)- Method to add products to the inventory (incoming products).
7. void sellProduct(String productName, int quantity, double sellingPrice)- Method to sell products from the inventory (outgoing products).
8. ElectronicProduct findProductByName(String name)-Method to find a product in the inventory by name.
9. ElectronicProduct findProductByName(String name, String brand)-Overloaded method to find a product in the inventory by name and brand.

C)Package - EmployeeManagement

1. public interface Accountant

METHODS-

calculateSalary(Double baseSalary, Integer... numberOfProductsSold)- Method to calculate the salary of an employee based on base salary and number of products sold.

1. class Employee implements EmployeeIncentives, Accountant

VARIABLES-

1. name- to store the name of the employee.

2. salary- salary given to employee.

3. leavesTaken- total number of leaves an employee has taken.

4. email- email of the employee.

5. phoneNumber- phone number of the employee.

6. joiningDate and exitDate- Date class variable to monitor joining and exit date.

METHODS-

1. String getName()- Getter and setter methods for name.
2. Date getJoiningDate()- Getter and setter methods for joining date and exit date.
3. Date getExitDate()- Getter for exit date.
4. void setExitDate()- Setter for exit date.
5. void setJoiningDate(Date joiningDate)- setter for joining date to “joiningDate”.
6. void displayEmployeeDetails()- Method to display employee details.
7. Double calculateSalary(Double baseSalary, Integer... numberOfProductsSold)- Method to calculate salary incorporating incentives.

3) private class ContactInfo- Nested class for contact information.

1. String getEmail() and void setEmail(String email)- getter and setter for email.
2. String getPhoneNumber()- Getter for phone number.
3. void setPhoneNumber(String phoneNumber)- Method to set phone number using nested ContactInfo class.
4. void displayContactInfo()- Method to display contact information.
5. Double calculateIncentives(Integer... numberOfProductsSold)- Method to calculate incentives based on the number of products sold.
6. Double calculateIncentives(Double baseIncentive, Integer... numberOfProductsSold)-Varargs overloading for calculating incentives with a custom base incentive value.

4) public interface EmployeeIncentives

METHODS-

1. Double calculateIncentives(Integer... numberOfProductsSold)-Method to calculate incentives based on the number of products sold.

5) public class MeetingThread implements Runnable

METHODS-

1. void run()-Run method to simulate attending a meeting.