

Q1 An inventory management system needs to track various types of products in a store: Book, Electronics, and Clothing. Each product type has a unique method to calculate its final price based on the base price, with Book having a fixed discount rate, Electronics having an additional warranty fee, and Clothing having a seasonal discount. You need to apply the polymorphism concept to calculate the final price.

You need to code only for these questions.

1) An abstract base class 'Product' with private attribute of `base_price` and a pure virtual function `calculateFinalPrice()` that calculates the price based on the product type.

2) Code for Derived classes:

'Book' with private member `discount_rate`,
'Electronics' with private member `warranty_fee`,
'Clothing' with a private member `seasonal_discount`.

Each class overrides the `calculateFinalPrice()` method with specific logic. You need to code for the `calculateFinalPrice()` method in each class and desired constructor and destructor.

3) A main function where you create an array (or a vector) of pointers to `Product`. Populate this array with instances of `Book`, `Electronics`, and `Clothing`. Then, iterate over this array, calling the `calculateFinalPrice()` method on each `Product`.

class Product {

private double base_price;

public:

virtual double calculateFinalPrice() = 0 { };

public: Product();

double discount_rate;

public:

Book(double a){ discount_rate=a; }

double b;

base_price=b;

double calculateFinalPrice(){

double result = base_price - discount_rate / 100;

return result; }

80%

class Electronics : public Product {

double warranty_Fee;

public:

double calculateFinalPrice(){

double result = base_price + warranty_Fee;

return result; }

Object Oriented Programming

SE102 T-B-Spring 2024

UBS IITU INFORMATION TECHNOLOGY UNIVERSITY

Electronics(double a, double b){ warranty_Fee=a; base_price=b; } };

class clothing : public Product {

 double seasonal-discount;

public:

 clothing (double a, double b) {
 seasonal-discount = a;
 base-price = b; }

 double calculateFinalPrice () {

 double result = base-price - seasonal-discount;
 return result; } }

int main () {

 vector<Product> p;

 p.pushback(clothing(10, 10));

 p.pushback(new Book(50, 0.5));

 p.pushback(new Electronic(60, 30));

 p.pushback(new Clothing(100, 0.3));

 for (Product* prd : p) {

 cout << prd->calculateFinalPrice() << endl;

 for (Product* prd : p) {

 delete prd; }

 return 0; }