

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;
};

class Book {
    private: double discount-rate;
    public:
        Book(double a) { discount-rate = a; base-price = b; }
        double calculateFinalPrice() {
            double result = base-price -  $\frac{\text{discount-rate}}{100}$ ;
            return result;
        }
};

class Electronics : public Product {
    private: double warranty-fee;
    public:
        Electronics(double a, double b) { warranty-fee = a; base-price = b; }
        double calculateFinalPrice() {
            double result = base-price + warranty-fee;
            return result;
        }
};
    
```

805

```

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

```

```

    new Book (50, 0.5));
    p.push_back (new Electronic (60, 30));
    p.push_back (new Clothing (100, 0.3));

```

```

for (Product *prd : p) {

```

```

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

```

```

for (Product *prt : p) {
    delete prt; }

```

```

return 0; }

```