Name : A.Karthickkannan

Reg.No : 2022506023

Subject : Advanced Data Structures & Laboratory

Subject code : IT5412

Title : Implementation of Pet shop Management System using OOPS Concepts

Date: 24/04/24

Implementation of Pet Shop Management System Using OOPS Concepts

Aim:

To implement Pet shop management system using OOPS Concepts in a unified application.

Description :

Pet Shop Management System & It's Functionalities:

1.Inventory Management:
The application provides comprehensive inventory management capabilities for a pet shop, allowing the addition of various items such as pets, products, and accessories.

2.Item Addition:
Users can add pets, products, and accessories to the inventory. For each item, details such as ID, name, type, quantity, price, and specific attributes (like breed for pets, manufacturer for products, and size/color for accessories) are recorded.

3.Order Management:
Customers can place orders for items available in the inventory. The system supports adding items to a cart, specifying quantities, and calculating the total order amount.

4.Wallet Integration:
A wallet feature is integrated, enabling users to deposit funds for making purchases. Transactions are tracked, allowing users to manage their balance effectively.

5.Checkout Process:
Customers can proceed to checkout, where the total order amount is calculated, and payment is processed using the funds available in the wallet. The system ensures that the wallet balance is sufficient before completing the transaction.

6.Order Tracking:
Once an order is placed, users can track their orders by entering the order ID. Details such as the items in the order, total amount, date, and status are displayed.

7.Search Functionality:
The application offers search functionality, allowing users to find specific items, such as pets or toys, by providing relevant details like ID, name, or type.

8.Error Handling:
Robust error handling mechanisms are implemented to handle scenarios such as item not found or insufficient funds, ensuring a smooth user experience.

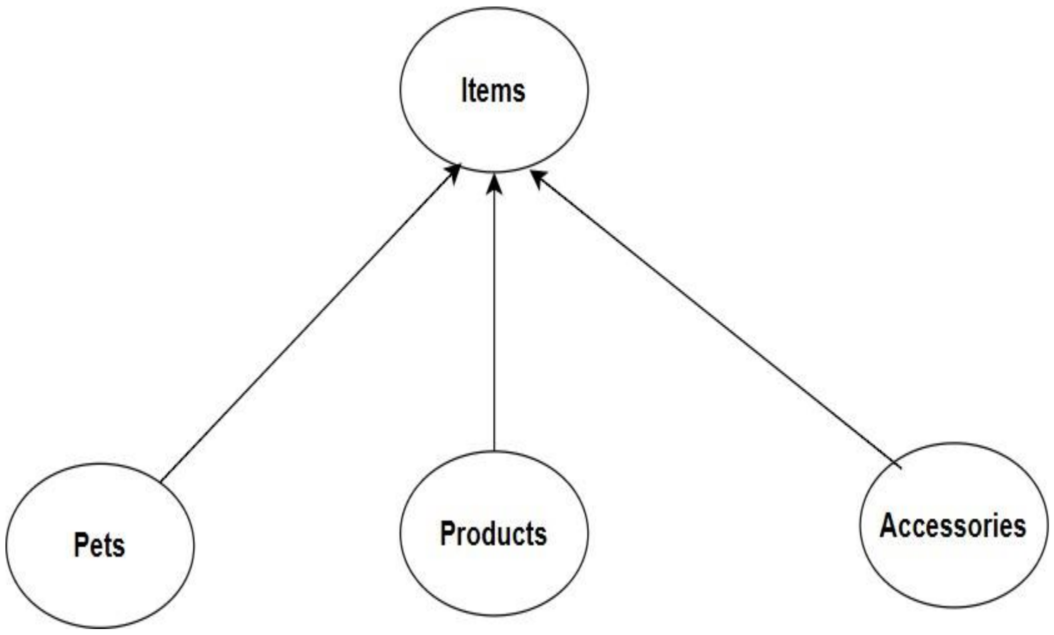
9.User Interface:
The application provides a user-friendly interface with a menu-driven system, guiding users through various functionalities and operations seamlessly.

The application is a Pet shop Management System implemented in C++. It allows users to manage inventory, add pets, products, and accessories, place orders, display inventory, check out orders, and perform wallet transactions. The system supports various features such as adding items to inventory, displaying inventory, ordering items, checking out orders, displaying total order price, adding money to the wallet, and finding pets or pet toys based on user input. It utilizes classes and inheritance to organize and manage different types of items such as pets, products, and accessories. Additionally, it includes features like wallet management for financial transactions and order tracking.

Source Code:

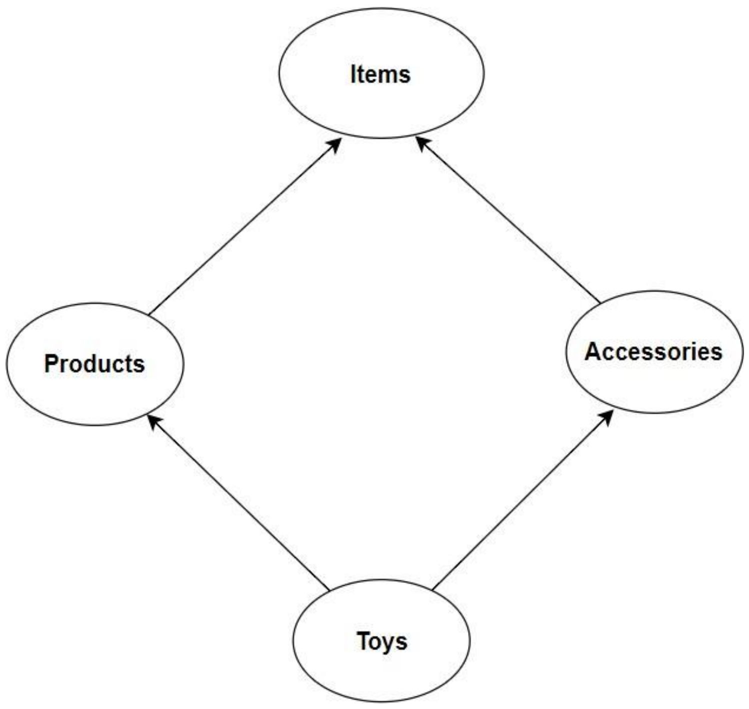
```
#include<iostream>
#include<vector>
#include<string>
#include<stdexcept>
using namespace std;
class Items {
protected:
    int Id, StockQty;
    float price;
    string name, type;

public:
    Items(): Id(0), name(""), type(""), StockQty(0), price(0.0) {}
    Items(int Id, string name, string type, int StockQty, float price): Id(Id),
name(name), type(type), StockQty(StockQty), price(price) {}

    virtual int getId() const = 0;
    virtual string getname() const = 0;
    virtual string gettype() const = 0; //Pure virtual functions
    virtual float getprice() const = 0;
    virtual int getStk() const = 0;
    virtual int updateStock(int qty) { StockQty = qty; return StockQty; }
    virtual void display() const = 0;
};
class Pets: public Items { //single inheritance
protected:
    int age;
    string breed, gender, color;
```

SINGLE INHERITANCE & HIERARCHIAL INHERITANCE IN PSMS



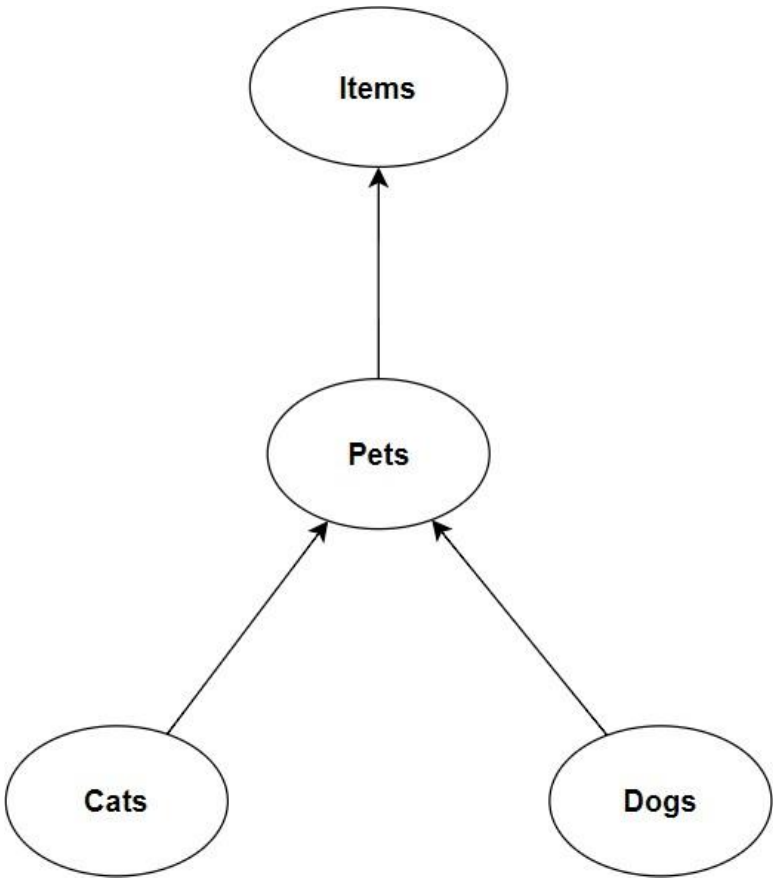
```
public:
        Pets(): Items(0,"","",0,0.0), breed(""), age(0), gender(""), color("") {}
        Pets(int Id, string name, string type, string breed, int age, string gender, string color,
    int StockQty, float price): Items(Id, name, type, StockQty, price), breed(breed),
    age(age), gender(gender), color(color) {}

        int getId() const override { return Id; }
        string getname() const override { return name; }
        string gettype() const override { return type; }
        float getprice() const override { return price; }
        int getStk() const override { return StockQty; }  //virtual functions
        int updateStock(int qty) override { StockQty = qty; return StockQty; }

        string getBreed() const { return breed; }

        void display() const override {
            cout << "Pet ID :" << Id << endl;
            cout << "Pet Name :" << name << endl;
            cout << "Type :" << type << endl;
            cout << "Breed :" << breed << endl;
            cout << "Age :" << age << endl;
            cout << "Gender :" << gender << endl;
            cout << "Color :" << color << endl;
            cout << "Stock Qty :" << StockQty << endl;
            cout << "Price :" << price << endl;
        }
    };
    class Products: public Items{ //hierarchial inheritance
    protected:
        string manfctr, Expdate;

    public:
        Products(): Items(0, "", "", 0, 0.0), manfctr(""), Expdate("") {}
        Products(int Id, string name, string type, string manfctr, string Expdate, int
    Stockqty, float price): Items(Id, name, type, StockQty, price), manfctr(manfctr),
    Expdate(Expdate) {}

        int getId() const override { return Id; }
        string getname() const override { return name; }
        string gettype() const override { return type; }
        int getStk() const override { return StockQty; }
        float getprice() const override { return price; }

        string getmanftr() const { return manfctr; }
```

MULTIPLE INHERITANCE & MULTIPATH INHERITANCE IN PSMS



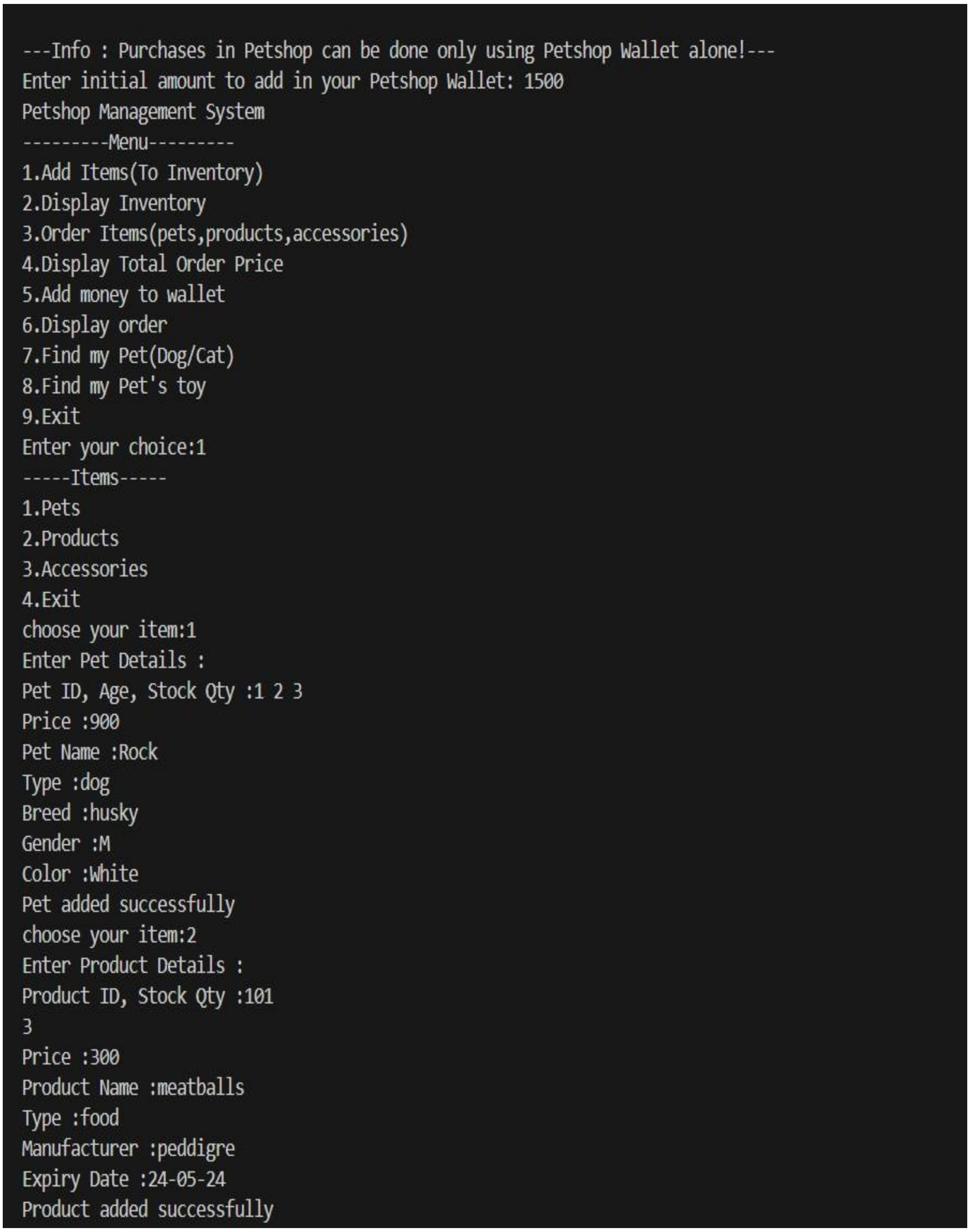
```
void setStockQty(int qty){StockQty = qty;}
    int updateStock(int qty) override { StockQty = qty; return StockQty; }

    void display() const override {
        cout << "Product ID :" << Id << endl;
        cout << "Product Name :" << name << endl;
        cout << "Type :" << type << endl;
        cout << "Manufacturer :" << manfctr << endl;
        cout << "Expiry Date :" << Expdate << endl;
        cout << "Stock Qty :" << StockQty << endl;
        cout << "Price :" << price << endl;
    }
};
class Accessories: public Items { //hierarchial inheritance
protected:
    string size, color;

public:
    Accessories(): Items(0, "", "", 0, 0.0), size(""), color("") {}
    Accessories(int Id, string name, string type, string size, string color, int Stockqty,
float price): Items(Id, name, type, StockQty, price), size(size), color(color) {}

    int getId() const override { return Id; }
    string getname() const override { return name; }
    string gettype() const override { return type; }
    float getprice() const override { return price; }
    int getStk() const override { return StockQty; }
    void setStockQty(int qty){StockQty = qty;}
    int updateStock(int qty) override { StockQty = qty; return StockQty; }

    string getsize() const { return size; }

    void display() const override {
        cout << "Accessory ID :" << Id << endl;
        cout << "Accessory Name :" << name << endl;
        cout << "Type :" << type << endl;
        cout << "Size :" << size << endl;
        cout << "Color :" << color << endl;
        cout << "Stock Qty :" << StockQty << endl;
        cout << "Price :" << price << endl;
    }
};
class Toys: public Products, public Accessories { //multiple & multipath inherritance
public:
```

MULTILEVEL INHERITANCE & HYBRID INHERITANCE IN PSMS



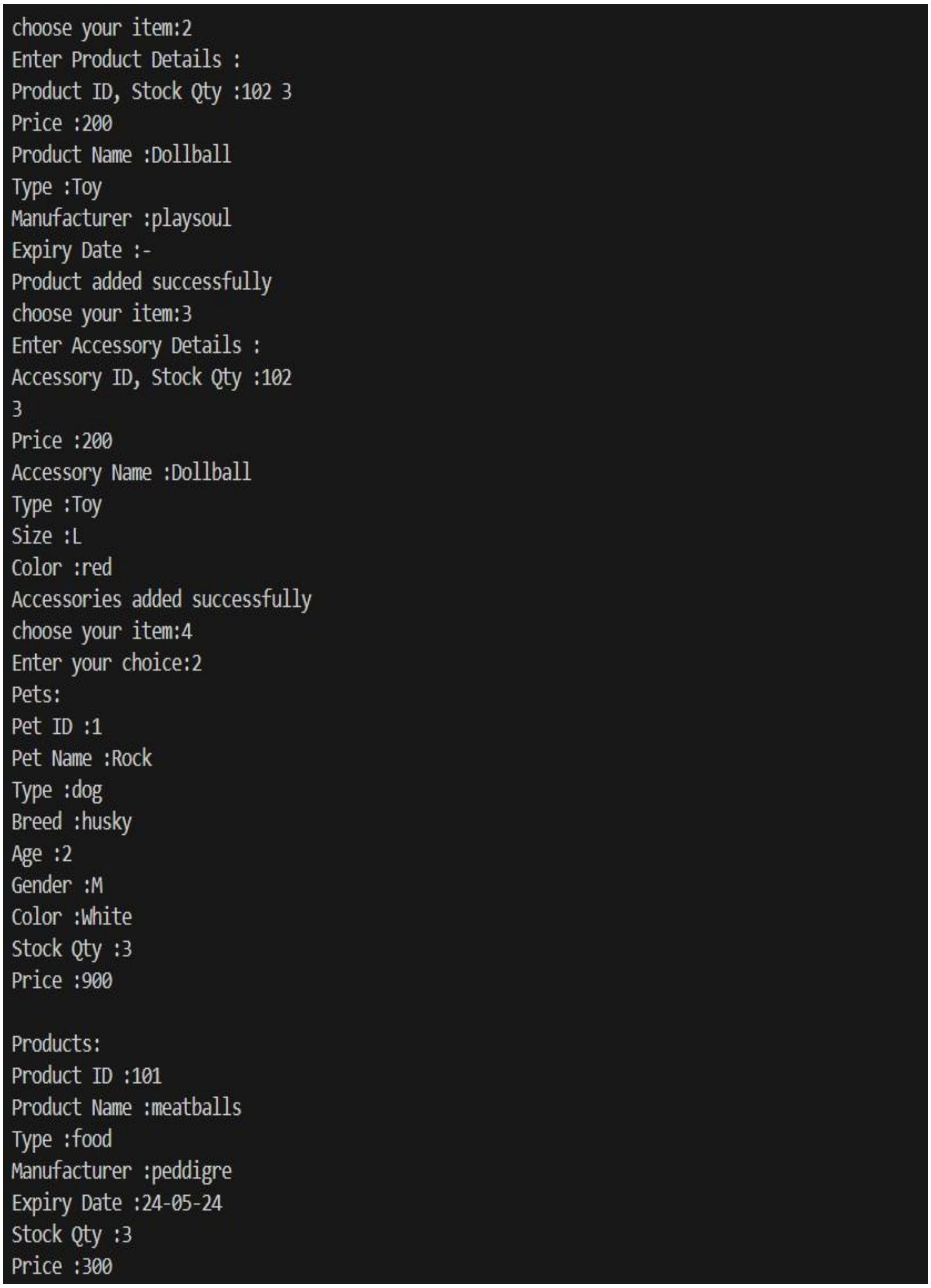
```
    Toys(int Id, string name, string manfctr, int StockQty, float price): Products(Id,
name, "Toy", manfctr, "-", StockQty, price), Accessories(Id, name, "Toy", "", "",
StockQty, price) {}

    void display() const override {
        cout << "Toy ID :" << Products::Id << endl;
        cout << "Toy Name :" << Products::name << endl;
        cout << "Manufacturer :" << Products::manfctr << endl;
        cout << "Size :" << Accessories::size << endl;
        cout << "Color :" << Accessories::color << endl;
        cout << "Stock Qty :" << Products::StockQty << endl;
        cout << "Price :" << Products::price << endl;
    }

    void setColsize(const string& tsize, const string& tcolor) {
        Accessories::size = tsize;
        Accessories::color = tcolor;
    }
};
class Dogs: public Pets { //multilevel inhertance
public:
    Dogs(int Id, string name, string breed, int age, string gender, string color, int
StockQty, float price): Pets(Id, name, "Dog", breed, age, gender, color, StockQty,
price) {}

    void display() const override {
        cout << "Dog ID :" << Id << endl;
        cout << "Dog Name :" << name << endl;
        cout << "Breed :" << breed << endl;
        cout << "Age :" << age << endl;
        cout << "Gender :" << gender << endl;
        cout << "Color :" << color << endl;
        cout << "Stock Qty :" << StockQty << endl;
        cout << "Price :" << price << endl;
    }
};
class Cats: public Pets { //hybrid inheritance
public:
    Cats(int Id, string name, string breed, int age, string gender, string color, int
StockQty, float price): Pets(Id, name, "Cat", breed, age, gender, color, StockQty,
price) {}
    void display() const override {
        cout << "Cat ID :" << Id << endl;
        cout << "Cat Name :" << name << endl;
        cout << "Breed :" << breed << endl;
```

OUTPUT:



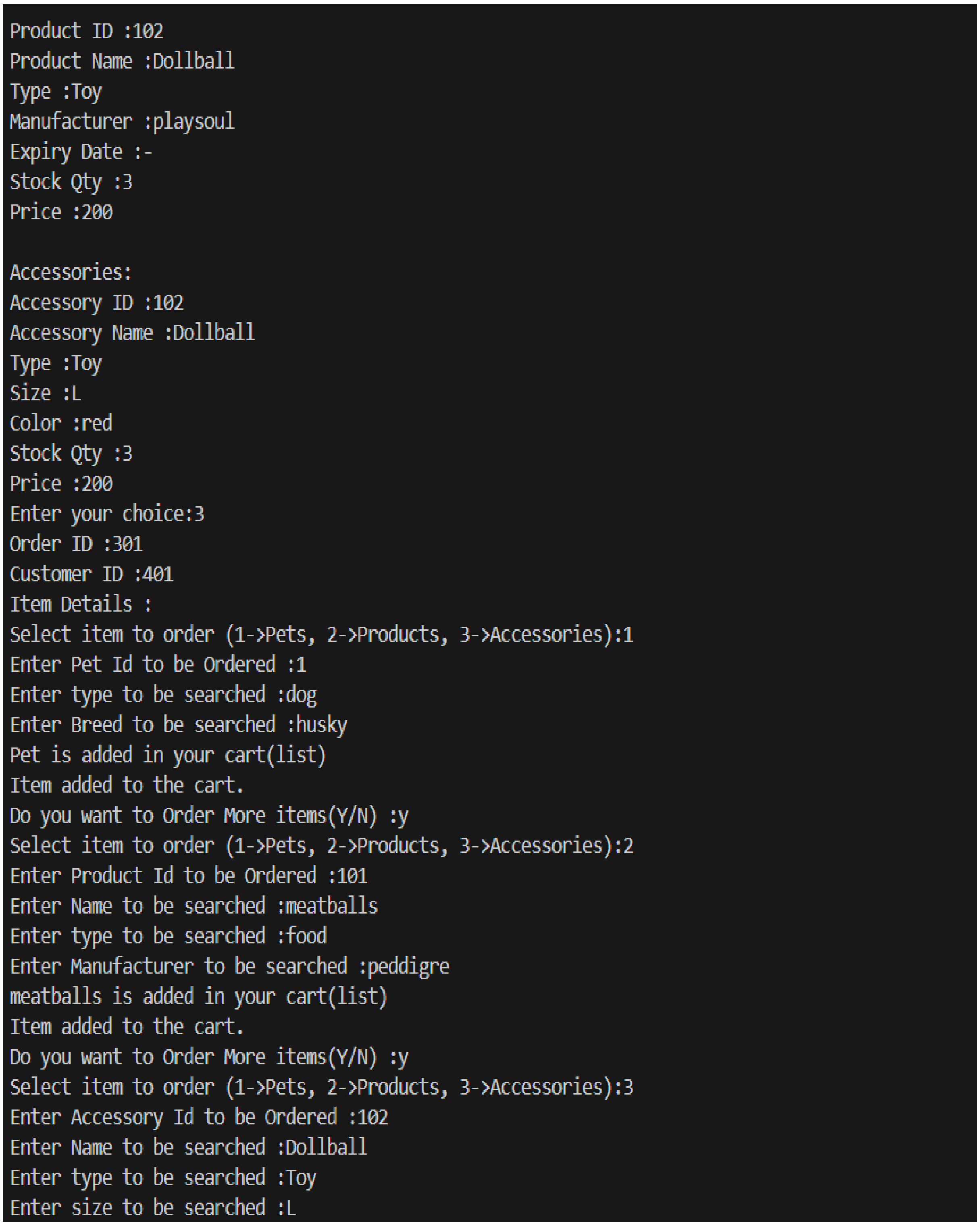
```
---Info : Purchases in Petshop can be done only using Petshop Wallet alone!---
Enter initial amount to add in your Petshop Wallet: 1500
Petshop Management System
---------Menu---------
1.Add Items(To Inventory)
2.Display Inventory
3.Order Items(pets,products,accessories)
4.Display Total Order Price
5.Add money to wallet
6.Display order
7.Find my Pet(Dog/Cat)
8.Find my Pet's toy
9.Exit
Enter your choice:1
-----Items-----
1.Pets
2.Products
3.Accessories
4.Exit
choose your item:1
Enter Pet Details :
Pet ID, Age, Stock Qty :1 2 3
Price :900
Pet Name :Rock
Type :dog
Breed :husky
Gender :M
Color :White
Pet added successfully
choose your item:2
Enter Product Details :
Product ID, Stock Qty :101
3
Price :300
Product Name :meatballs
Type :food
Manufacturer :peddigre
Expiry Date :24-05-24
Product added successfully
```

```
cout << "Age :" << age << endl;
        cout << "Gender :" << gender << endl;
        cout << "Color :" << color << endl;
        cout << "Stock Qty :" << StockQty << endl;
        cout << "Price :" << price << endl;
    }
};
class Cart {
public:
    int id, qty;
    string name, type;
    float pr;
    int getid() const{return id;}
    int getqty() const{return qty;}
    string getname() const{return name;}
    string gettype() const{return type;}
    float getprice() const{return pr;}

    Cart(): id(0), name(""), type(""), qty(0), pr(0.0) {}
    Cart(int id, string name, string type, int qty, float pr): id(id), name(name),
type(type), qty(qty), pr(pr) {}
    float getpr() const { return pr; }
    Cart& operator+=(const Cart& other){
        this->qty += other.qty;
        this->pr += other.pr;
            return *this;
        }
};
class Order: public Cart { //single inheritance
private:
    int OrderId, CustId;
    float TotAmount;
    string date, status;

public:
    vector<Cart> cart;

    Order(int OrderId, int CustId, float TotAmount, string date,string status):
OrderId(OrderId), CustId(CustId), TotAmount(TotAmount), date(date), status(status)
{}
    int getOId() const { return OrderId; }
    int getCId() const { return CustId; }
    float getTot() const { return TotAmount; }
    string getdate() const { return date; }
    string getstat() const { return status; }
```



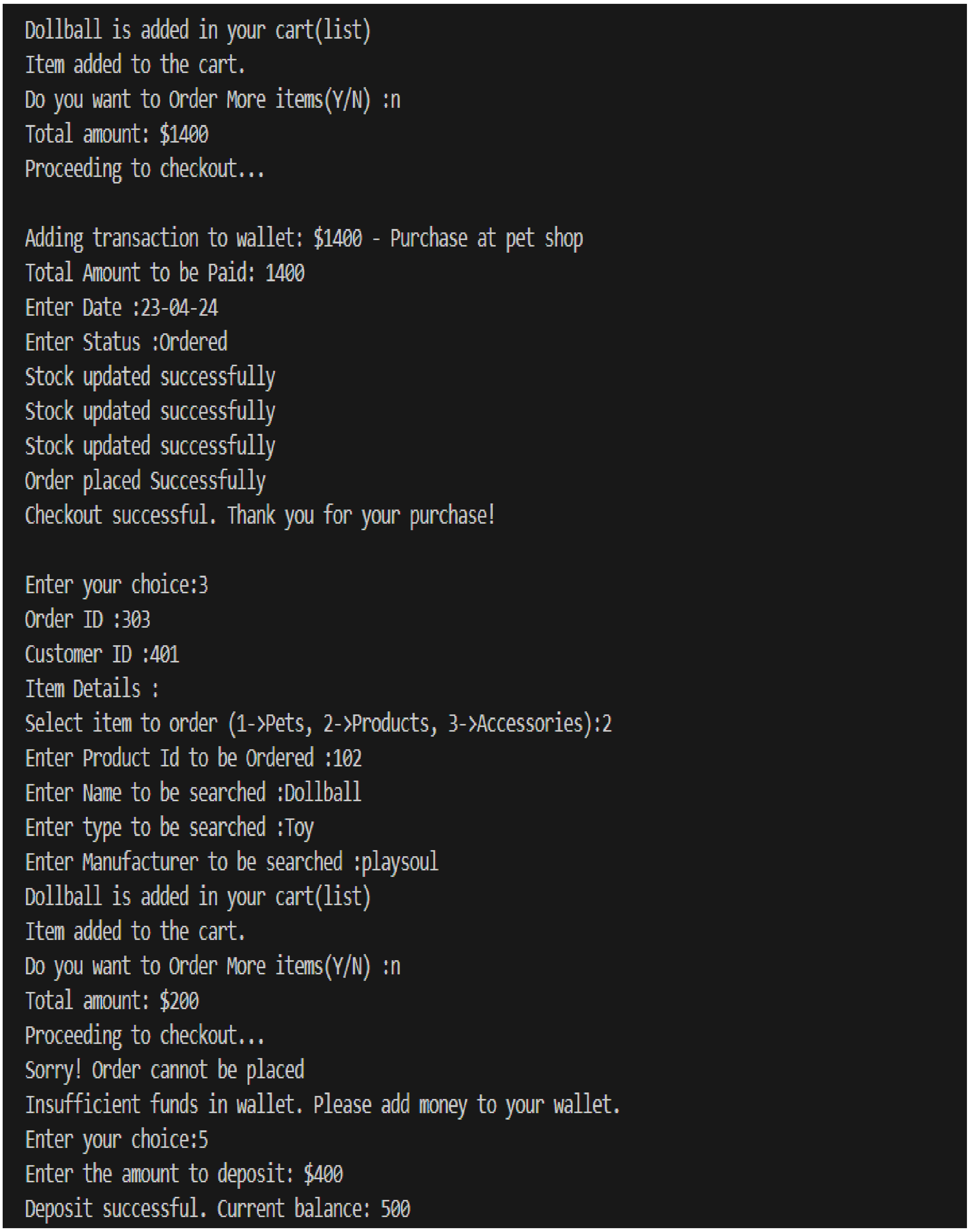
```
choose your item:2
Enter Product Details :
Product ID, Stock Qty :102 3
Price :200
Product Name :Dollball
Type :Toy
Manufacturer :playsoul
Expiry Date :-
Product added successfully
choose your item:3
Enter Accessory Details :
Accessory ID, Stock Qty :102
3
Price :200
Accessory Name :Dollball
Type :Toy
Size :L
Color :red
Accessories added successfully
choose your item:4
Enter your choice:2
Pets:
Pet ID :1
Pet Name :Rock
Type :dog
Breed :husky
Age :2
Gender :M
Color :White
Stock Qty :3
Price :900

Products:
Product ID :101
Product Name :meatballs
Type :food
Manufacturer :peddigre
Expiry Date :24-05-24
Stock Qty :3
Price :300
```

```
  void Addtocart(int i, const string& n, const string& t, int Qty, float pr) {
        cart.push_back(Cart(i, n, t, Qty, pr)); // Instantiate Cart directly with non-abstract
constructor
        cout << n << " is added in your cart(list)" << endl;
    }
    void setAmount(float amount) {
        TotAmount = amount;
    }
    void setDate(const string& Ordate) {
        date = Ordate;
    }
    void setStatus(const string& OrStatus) {
        status = OrStatus;
    }
    float CalcTotal() const{
            Cart citems;
            for (auto& item : cart){
                citems += item;
            }
            return citems.getpr();
        }
};
template <typename I,typename S,typename F>
class Transaction{
private:
    F amount;
    S description;
public:
    Transaction(F amount,S description): amount(amount),description(description){}
     F getAmount() const {return amount;}
     S getDesc() const{return description;}
};
template <typename I,typename S,typename F>
class Wallet{
private:
    F balance;
    vector<Transaction<I,S,F>> transactions;
public:
    Wallet():balance(0.0){}
    Wallet(F initialAmount):balance(initialAmount){}
    F getBalance() const{return balance;}
    void deposit(F amount){
        balance += amount;
        cout<< "Deposit successful. Current balance: "<< balance <<endl;
     }
```



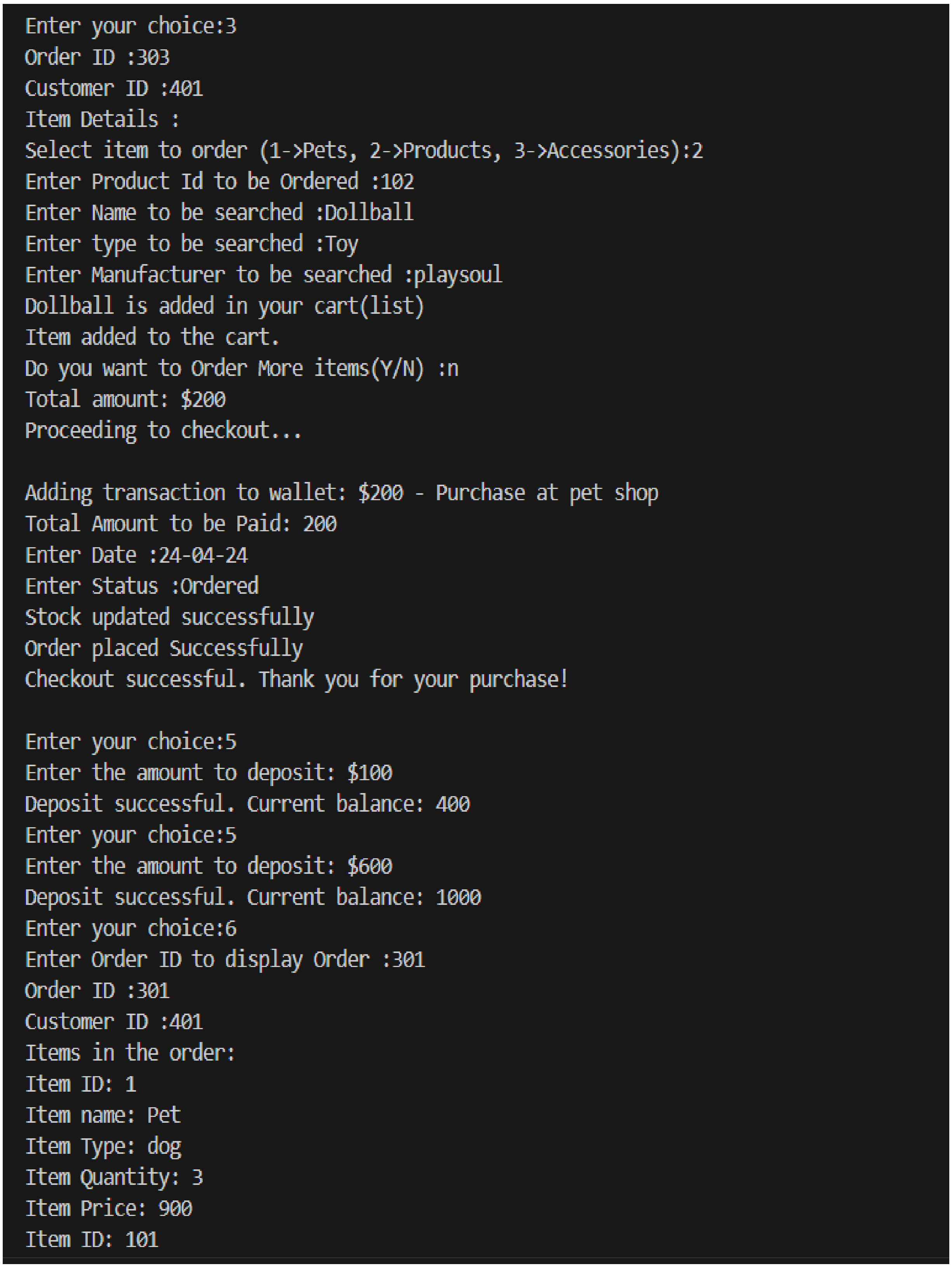
```
Product ID :102
Product Name :Dollball
Type :Toy
Manufacturer :playsoul
Expiry Date :-
Stock Qty :3
Price :200

Accessories:
Accessory ID :102
Accessory Name :Dollball
Type :Toy
Size :L
Color :red
Stock Qty :3
Price :200
Enter your choice:3
Order ID :301
Customer ID :401
Item Details :
Select item to order (1->Pets, 2->Products, 3->Accessories):1
Enter Pet Id to be Ordered :1
Enter type to be searched :dog
Enter Breed to be searched :husky
Pet is added in your cart(list)
Item added to the cart.
Do you want to Order More items(Y/N) :y
Select item to order (1->Pets, 2->Products, 3->Accessories):2
Enter Product Id to be Ordered :101
Enter Name to be searched :meatballs
Enter type to be searched :food
Enter Manufacturer to be searched :peddigre
meatballs is added in your cart(list)
Item added to the cart.
Do you want to Order More items(Y/N) :y
Select item to order (1->Pets, 2->Products, 3->Accessories):3
Enter Accessory Id to be Ordered :102
Enter Name to be searched :Dollball
Enter type to be searched :Toy
Enter size to be searched :L
```

```
    void withdraw(F amount){
        if(balance >= amount){
            balance-=amount;
            cout<<""<<endl;}
        else{cout<<"Insufficient funds in wallet!"<<endl;}}
    void addTransaction(F amount, S& description) {
        cout << "Adding transaction to wallet: $" << amount << " - " << description <<
endl;
        transactions.push_back(Transaction<I,S,F>(amount,description));
}
};


class Inventory {
protected:
    vector<Pets> pets;
    vector<Products> products;
    vector<Accessories> accessories;
    vector<Dogs> dogs;
    vector<Cats> cats;
    vector<Toys> toys;
public:
    vector<Order> orders;
    float tot;
    void addpet(int Id, string name, string type, string breed, int age, string gender,
string color, int StockQty, float price) {
        pets.emplace_back(Pets(Id, name, type, breed, age, gender, color, StockQty,
price));
        cout << "Pet added successfully" << endl;
        if (type == "Dog") {
            dogs.push_back(Dogs(Id, name, breed, age, gender, color, StockQty, price));
        } else if (type == "Cat") {
            cats.push_back(Cats(Id, name, breed, age, gender, color, StockQty, price));
        }
    }
    void addproduct(int Id, string name, string type, string manfctr, string Expdate, int
StockQty, float price) {
        Products prod(Id, name, type, manfctr, Expdate, StockQty, price);
        prod.setStockQty(StockQty);
        products.emplace_back(prod);
        cout << "Product added successfully" << endl;
        if (type == "Toy") {
            toys.push_back(Toys(Id, name, manfctr, StockQty, price));
        }
    }
```



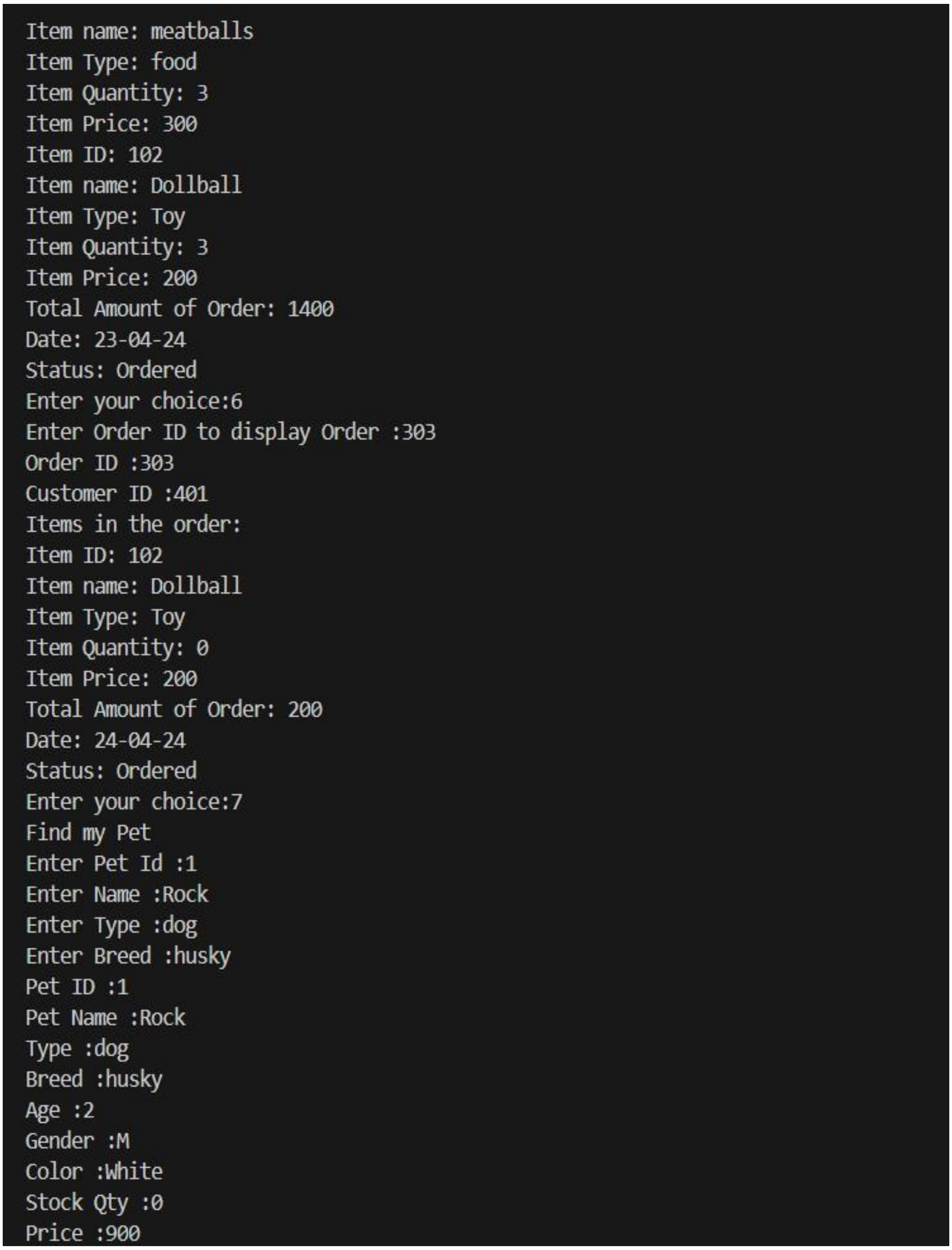
```
Dollball is added in your cart(list)
Item added to the cart.
Do you want to Order More items(Y/N) :n
Total amount: $1400
Proceeding to checkout...

Adding transaction to wallet: $1400 - Purchase at pet shop
Total Amount to be Paid: 1400
Enter Date :23-04-24
Enter Status :Ordered
Stock updated successfully
Stock updated successfully
Stock updated successfully
Order placed Successfully
Checkout successful. Thank you for your purchase!

Enter your choice:3
Order ID :303
Customer ID :401
Item Details :
Select item to order (1->Pets, 2->Products, 3->Accessories):2
Enter Product Id to be Ordered :102
Enter Name to be searched :Dollball
Enter type to be searched :Toy
Enter Manufacturer to be searched :playsoul
Dollball is added in your cart(list)
Item added to the cart.
Do you want to Order More items(Y/N) :n
Total amount: $200
Proceeding to checkout...
Sorry! Order cannot be placed
Insufficient funds in wallet. Please add money to your wallet.
Enter your choice:5
Enter the amount to deposit: $400
Deposit successful. Current balance: 500
```

```
void addaccessory(int Id, string name, string type,const string& size,const string&
color, int StockQty, float price) {
        Accessories access(Id, name, type, size, color, StockQty, price);
        access.setStockQty(StockQty);
        accessories.emplace_back(access);
        cout << "Accessories added successfully" << endl;
        if (type == "Toy") {
            for (auto& toy : toys) {
                if (toy.Products::getId() == Id) {
                    toy.setColsize(size, color);
                }
            }
        }
    }
    /*void petdisp() const {
        for(const auto& pet : pets) {
            pet.display();
        }
    }
    void productdisp() const {
        for(const auto& product : products) {
            product.display();
        }
    }

    void accessdisp() const {
        for(const auto& accessory : accessories) {
            accessory.display();
        }
    }*/

    const Pets& getpet(int Id, string& name, string& type, string& breed) const {
        for(const auto& pet : pets) {
            if(pet.getId() == Id && pet.getname() == name && pet.gettype() == type &&
pet.getBreed() == breed) {
                return pet;
        }
        throw runtime_error("Pet not found");
    }}
    const Pets& getpet(int Id, string& type, string& breed) const{
        for(const auto& pet : pets) {
            if(pet.getId() == Id && pet.gettype() == type && pet.getBreed() == breed) {
                return pet;
        }
        throw runtime_error("Pet not found"); }}
```



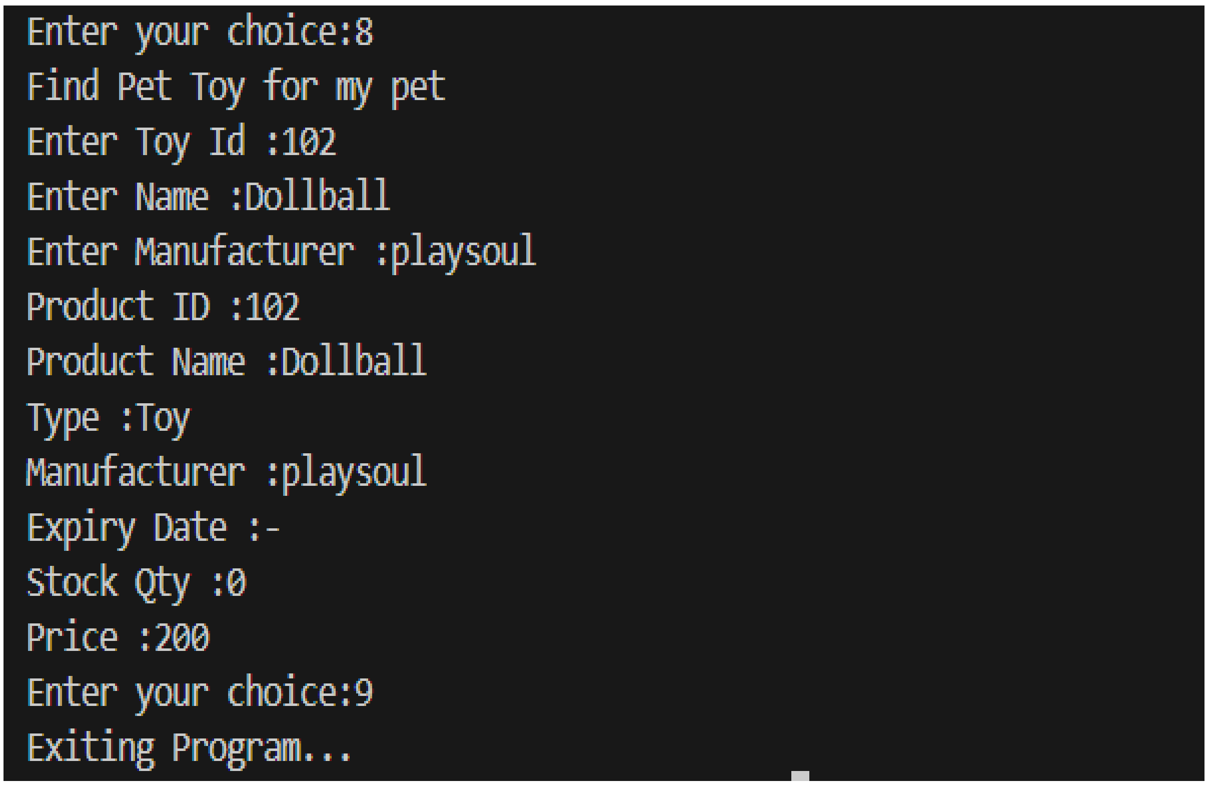
```
Enter your choice:3
Order ID :303
Customer ID :401
Item Details :
Select item to order (1->Pets, 2->Products, 3->Accessories):2
Enter Product Id to be Ordered :102
Enter Name to be searched :Dollball
Enter type to be searched :Toy
Enter Manufacturer to be searched :playsoul
Dollball is added in your cart(list)
Item added to the cart.
Do you want to Order More items(Y/N) :n
Total amount: $200
Proceeding to checkout...

Adding transaction to wallet: $200 - Purchase at pet shop
Total Amount to be Paid: 200
Enter Date :24-04-24
Enter Status :Ordered
Stock updated successfully
Order placed Successfully
Checkout successful. Thank you for your purchase!

Enter your choice:5
Enter the amount to deposit: $100
Deposit successful. Current balance: 400
Enter your choice:5
Enter the amount to deposit: $600
Deposit successful. Current balance: 1000
Enter your choice:6
Enter Order ID to display Order :301
Order ID :301
Customer ID :401
Items in the order:
Item ID: 1
Item name: Pet
Item Type: dog
Item Quantity: 3
Item Price: 900
Item ID: 101
```

```
    const Products& getproduct(int Id, string name, string type, string manfctr)const{
        for(const auto& product : products) {
            if(product.getId() == Id && product.getname() == name && product.gettype()
== type && product.getmanftr() == manfctr) {
                return product;
            }
        }
        throw runtime_error("Product not found");
    }
    const Accessories& getaccessory(int Id, string name, string type, string size)const{
        for(const auto& accessory : accessories) {
            if(accessory.getId() == Id && accessory.getname() == name &&
accessory.gettype() == type && accessory.getsize() == size) {
                return accessory;
            }
        }
        throw runtime_error("Accessory not found");
    }
    void displayInv() const {
        cout << "Pets:" << endl;
        for (const auto& p : pets) {
            p.display();
        }
        cout << "\nProducts:" << endl;
        for (const auto& pr : products) {
            pr.display();
        }
        cout << "\nAccessories:" << endl;
        for (const auto& acc : accessories) {
            acc.display();
        }
    }
    void dispOrder(int OrdId) const{
        bool found = false;
        for(const auto& ord: orders){
        if(ord.getOId() == OrdId){
            found = true;
        cout<<"Order ID :"<<ord.getOId()<<endl;
        cout<<"Customer ID :"<<ord.getCId()<<endl;
        cout<<"Items in the order:" <<endl;
        for(const auto& item : ord.cart){
            cout<<"Item ID: "<<item.getid()<<endl;
            cout<<"Item name: "<<item.getname()<<endl;
            cout<<"Item Type: "<<item.gettype()<<endl;
            cout<<"Item Quantity: "<<item.getqty()<<endl;
```



```
Item name: meatballs
Item Type: food
Item Quantity: 3
Item Price: 300
Item ID: 102
Item name: Dollball
Item Type: Toy
Item Quantity: 3
Item Price: 200
Total Amount of Order: 1400
Date: 23-04-24
Status: Ordered
Enter your choice:6
Enter Order ID to display Order :303
Order ID :303
Customer ID :401
Items in the order:
Item ID: 102
Item name: Dollball
Item Type: Toy
Item Quantity: 0
Item Price: 200
Total Amount of Order: 200
Date: 24-04-24
Status: Ordered
Enter your choice:7
Find my Pet
Enter Pet Id :1
Enter Name :Rock
Enter Type :dog
Enter Breed :husky
Pet ID :1
Pet Name :Rock
Type :dog
Breed :husky
Age :2
Gender :M
Color :White
Stock Qty :0
Price :900
```

```
          cout<<"Item Price: "<<item.getprice()<<endl;
        }
        cout<<"Total Amount of Order: "<<ord.getTot()<<endl;
        cout<<"Date: "<<ord.getdate()<<endl;
        cout<<"Status: "<<ord.getstat()<<endl;
        break;}
         }
         if(!found){cout<<"Order not found!"<<endl;}
     }
    int checkout(float tot, Wallet<int,string,float>& userWallet,string& stat){
            cout << "Total amount: $" << tot << endl;
            cout << "Proceeding to checkout..." << endl;
            if (userWallet.getBalance() >= tot) {
                userWallet.withdraw(tot);
                userWallet.addTransaction(tot, stat);
                return 2;
            } else { return -1;}
    }
    void removeItemFromInventory(int Id, string type, int qty) {
    if (type == "Pet" || type == "dog" || type == "cat" || type == "bird" || type == "fish" ||
type == "small-mammals" || type == "reptiles") {
        for (auto& item : pets) {
            if (item.getId() == Id) {
                item.updateStock(item.getStk() - qty);
                cout << "Stock updated successfully" << endl;
                return;
            }
        }
    } else if (type == "Product" || type == "food" || type == "Toy" || type == "beds" ||
type == "clothes" || type == "Apparel" || type == "litter") {
        for (auto& item : products) {
            if (item.getId() == Id) {
                item.updateStock(item.getStk() - qty);
                cout << "Stock updated successfully" << endl;
                return;
            }
        }
    } else if (type == "Accessory" || type == "Toy" || type == "harnesses" || type ==
"collars&leashes" || type == "grooming") {
        for (auto& item : accessories) {
            if (item.getId() == Id) {
                item.updateStock(item.getStk() - qty);
                cout << "Stock updated successfully" << endl;
                return;
            }  }
```



```
Enter your choice:8
Find Pet Toy for my pet
Enter Toy Id :102
Enter Name :Dollball
Enter Manufacturer :playsoul
Product ID :102
Product Name :Dollball
Type :Toy
Manufacturer :playsoul
Expiry Date :-
Stock Qty :0
Price :200
Enter your choice:9
Exiting Program...
```

```
 }
   throw runtime_error("Item not found in inventory");
}
void updateStockAfterOrder(vector<Cart>& cart) {
   for (const auto& item : cart) {
      try {
         removeItemFromInventory(item.getid(), item.gettype(), item.getqty());
      } catch (const runtime_error& e) {
         cerr << "Error: " << e.what() << endl;
      }
   }
}
};
int main() {
   cout<<"---Info : Purchases in Petshop can be done only using Petshop Wallet
alone!---"<<endl;
   Inventory invtry;
   float icost,dcost;
   cout<<"Enter initial amount to add in your Petshop Wallet: ";
   cin>>icost;
   Wallet<int,string,float> wallet(icost);
   char ch;
   int c, co;
   int Id, age, StockQty;
   string name, type, breed, gender, color, manfctr, Expdate, size;
   string st = "Purchase at pet shop";
   float price,tot=0.0;
   int OrderId, CustId;
   string date, status;
   cout << "Petshop Management System"<<endl;
   cout << "---------Menu --------" << endl;
   cout << "1.Add Items(To Inventory)" << endl;
   cout << "2.Display Inventory" << endl;
   cout << "3.Order Items(pets,products,accessories)" << endl;
   cout << "4.Display Total Order Price" << endl;
   cout << "5.Add money to wallet" << endl;
   cout << "6.Display order" << endl;
   cout << "7.Find my Pet(Dog/Cat)" << endl;
   cout << "8.Find my Pet's toy" << endl;
   cout << "9.Exit"<<endl;
   while (1) {
      cout << "Enter your choice:";
      cin >> c;
      if (c==9){cout<<"Exiting Program...\n"; break;}
      switch (c) {
```

```
case 1:
     cout << "-----Items ---- " << endl;
     cout << "1.Pets" << endl;
     cout << "2.Products" << endl;
     cout << "3.Accessories" << endl;
     cout<< "4.Exit" << endl;
     do {
         cout << "choose your item:";
         cin >> co;
         switch (co) {
             case 1: {
                 cout << "Enter Pet Details :" << endl;
                 cout << "Pet ID, Age, Stock Qty :";
                 cin >> Id >> age >> StockQty;
                 cout << "Price :";
                 cin >> price;
                 cin.ignore();
                 cout << "Pet Name :";
                 getline(cin, name);
                 cout << "Type :";
                 getline(cin, type);
                 cout << "Breed :";
                 getline(cin, breed);
                 cout << "Gender :";
                 getline(cin, gender);
                 cout << "Color :";
                 getline(cin, color);
                 invtry.addpet(Id, name, type, breed, age, gender, color, StockQty,
price);
                 break; }
             case 2: {
                 cout << "Enter Product Details :" << endl;
                 cout << "Product ID, Stock Qty :";
                 cin >> Id >> StockQty;
                 cout << "Price :";
                 cin >> price;
                 cin.ignore();
                 cout << "Product Name :";
                 getline(cin, name);
                 cout << "Type :";
                 getline(cin, type);
                 cout << "Manufacturer :";
                 getline(cin, manfctr);
                 cout << "Expiry Date :";
                 getline(cin, Expdate);
```

```
                    invtry.addproduct(Id, name, type, manfctr, Expdate, StockQty,
         price);
                break;
            }
            case 3: {
                cout << "Enter Accessory Details :" << endl;
                cout << "Accessory ID, Stock Qty :";
                cin >> Id >> StockQty;
                cout << "Price :";
                cin >> price;
                cin.ignore();
                cout << "Accessory Name :";
                getline(cin, name);
                cout << "Type :";
                getline(cin, type);
                cout << "Size :";
                getline(cin, size);
                cout << "Color :";
                getline(cin, color);
                invtry.addaccessory(Id, name, type, size, color, StockQty, price);
                break;
            }
        }
    } while (co < 4 && co > 0);
break;
case 2: {
    invtry.displayInv();
    break;
}

case 3: {
    cout << "Order ID :";
    cin >> OrderId;
    cout << "Customer ID :";
    cin >> CustId;
    cout << "Item Details :" << endl;
    Order norder(OrderId, CustId, 0.0, "", "");
    do {
        cout << "Select item to order (1->Pets, 2->Products, 3->Accessories):";
        cin >> co;
        if (co == 1) {
            cout << "Enter Pet Id to be Ordered :";
            cin >> Id;
            cin.ignore();
```

```
                cout << "Enter type to be searched :";
                getline(cin, type);
                cout << "Enter Breed to be searched :";
                getline(cin, breed);
                try {
                    const Pets& pet = invtry.getpet(Id,type, breed);
                    norder.Addtocart(Id, "Pet", type, pet.getStk(), pet.getprice());
                    cout << "Item added to the cart." << endl;
                } catch (const runtime_error& e) {
                    cerr << "Error: " << e.what() << endl;
                    cout << "Please try again." << endl;
                }
            }
            if (co == 2) {
                cout << "Enter Product Id to be Ordered :";
                cin >> Id;
                cin.ignore();
                cout << "Enter Name to be searched :";
                getline(cin, name);
                cout << "Enter type to be searched :";
                getline(cin, type);
                cout << "Enter Manufacturer to be searched :";
                getline(cin, manfctr);
                try {
                    const Products& product = invtry.getproduct(Id, name, type,
manfctr);
                    norder.Addtocart(Id, name, type, product.getStk(),
product.getprice());
                    cout << "Item added to the cart." << endl;
                } catch (const runtime_error& e) {
                    cerr << "Error: " << e.what() << endl;
                    cout << "Please try again." << endl;
                }
            }
            if (co == 3) {
                cout << "Enter Accessory Id to be Ordered :";
                cin >> Id;
                cin.ignore();
                cout << "Enter Name to be searched :";
                getline(cin, name);
                cout << "Enter type to be searched :";
                getline(cin, type);
                cout << "Enter size to be searched :";
                getline(cin, size);
                try {
```

```
                const Accessories& accessory = invtry.getaccessory(Id, name, type,
        size);
                    norder.Addtocart(Id, name, type, accessory.getStk(),
accessory.getprice());
                    cout << "Item added to the cart." << endl;
                } catch (const runtime_error& e) {
                    cerr << "Error: " << e.what() << endl;
                    cout << "Please try again." << endl;
                }
            }
            cout << "Do you want to Order More items(Y/N) :";
            cin >> ch;
        } while (ch == 'Y' || ch == 'y');
        int WallStat = invtry.checkout(norder.CalcTotal(),wallet,st);
        if(WallStat != -1){
            norder.setAmount(norder.CalcTotal());
            tot += norder.CalcTotal();
            cout << "Total Amount to be Paid: " << norder.CalcTotal() << endl;
            cout << "Enter Date :";
            cin >> date;
            cout << "Enter Status :";
            cin >> status;
            norder.setDate(date);
            norder.setStatus(status);
            invtry.orders.push_back(norder);
            invtry.updateStockAfterOrder(norder.cart);
            cout << "Order placed Successfully" << endl;
            cout << "Checkout successful. Thank you for your purchase!\n" << endl;
        }
        else{
            cout<<"Sorry! Order cannot be placed"<<endl;
            cout << "Insufficient funds in wallet. Please add money to your wallet."
<< endl; }
        break;
    }
      case 4: {
        cout << "Total Amount of Orders: " << tot << endl;
        break;
      }
      case 5:{
        cout << "Enter the amount to deposit: $";
        cin >> dcost;
        wallet.deposit(dcost);
        break; }
```

```
        case 6:{
            cout << "Enter Order ID to display Order :";
            cin>>OrderId;
            invtry.dispOrder(OrderId);
            break;
         }
         case 7: {
            cout << "Find my Pet" << endl;
            cout << "Enter Pet Id :";
            cin >> Id;
            cin.ignore();
            cout << "Enter Name :";
            getline(cin, name);
            cout << "Enter Type :";
            getline(cin, type);
            cout << "Enter Breed :";
            getline(cin, breed);
            try {
               const Pets& pet = invtry.getpet(Id, name, type, breed);
               pet.display();
            } catch (const runtime_error& e) {
               cerr << "Error: " << e.what() << endl;
               cout << "Pet not found." << endl; }
            break;}
         case 8: {
            cout << "Find Pet Toy for my pet" << endl;
            cout << "Enter Toy Id :";
            cin >> Id;
            cin.ignore();
            cout << "Enter Name :";
            getline(cin, name);
            cout << "Enter Manufacturer :";
            getline(cin, manfctr);
            try { const Products& product = invtry.getproduct(Id, name, "Toy",
manfctr);
               product.display();
            } catch (const runtime_error& e) {
               cerr << "Error: " << e.what() << endl;
               cout << "Toy not found." << endl; }
            break; }
         default:
            cout << "Invalid Input" << endl;
      break; } }
   return 0;
}
```

RESULT :

Thus, Pet shop management system have implemented using OOPS Concepts in a unified application.