Shop Simulator – Ett affärssystem i miniatyr

Den här övningen ger praktisk erfarenhet av:

- Att designa klasshierarkier med basklasser och subklasser (arv)
- Att använda `enum class` där det är rimligt
- Konstruktorer och konstruktorordning
- Objekt som interagerar via medlemsfunktioner och komposition

---

Scenario

Ni ska simulera en enkel affär där **kunder** kommer in och försöker **köpa produkter** enligt en shoppinglista.

- **Affären** har ett **lager av produkter** (`Item`)
- **Varje kund** har en viss summa pengar och en shoppinglista
- Om pengarna räcker, köper kunden produkten
- När varor köps, dras de från lagret och affärens **kassa ökar**
- Affären kan också köpa in nya produkter om lagret tar slut

Klassdesign: Obligatoriska klasser

`Item` (basklass)

- Attribut:
    - `std::string name`
    - `int price` (kr)
- Metod:
    - `printInfo()`

`FoodItem` och `ToolItem` (subklasser till `Item`)

- `FoodItem` har: `int calories`
- `ToolItem` har: `int durability`

Dessa visar arv, men används inte polymorft ännu.

`Customer`

- Attribut:
    - `std::string name`
    - `int money`
    - `std::vector<std::string> shoppingList`
- Metoder:
    - `tryToBuy(Shop&)` – försöker köpa alla produkter på listan

Shop

- Attribut:
    - `std::vector<Item*> inventory` (eller `std::vector<std::unique_ptr<Item>>`)
    - `int balance`
- Metoder:
    - `printInventory()`
    - `sellItemToCustomer(const std::string& itemName, Customer&)`
    - `restock(Item*)`

Kodskelett (för inspiration)

```
class Item {
protected:
    std::string name_;
    int price_;
public:
    Item(const std::string& name, int price)
        : name_{name}, price_{price} {}
    std::string name() const { return name_; }
    int price() const { return price_; }

    void printInfo() const {
        std::cout << name_ << " – " << price_ << " kr\n";
    }
};

class FoodItem : public Item {
    int calories_;
public:
    FoodItem(const std::string& name, int price, int calories)
        : Item(name, price), calories_{calories} {}
};

class Customer {
    std::string name_;
    int money_;
    std::vector<std::string> shoppingList_;
public:
    // ... konstruktor
    void tryToBuy(Shop& shop);
};
```

Uppgifter

Grundläggande:

1. Implementera alla klasser.
2. Lägg till minst 5 produkter i Shopens inventory.

3. Skapa 2 kunder med olika shoppinglistor och pengar.
4. Simulera ett besök – kunden försöker köpa varje vara.
5. Visa:
    - Vilka produkter som köptes
    - Vad som fanns kvar i inventory
    - Shopens nya balans

Diskussion:

- När bör `ItemType` vara en `enum class` istället för flera subklasser?
- Hade polymorfism gjort något enklare?
- Hur hanteras konstruktorordningen?

Bonusuppgifter:

- Skriv en metod `Shop::restockRandom()` som fyller på lagret slumpmässigt.
- Lägg till ett attribut "lagersaldo" per produkt.
- Implementera en “ShoppingDay” där flera kunder handlar i turordning.