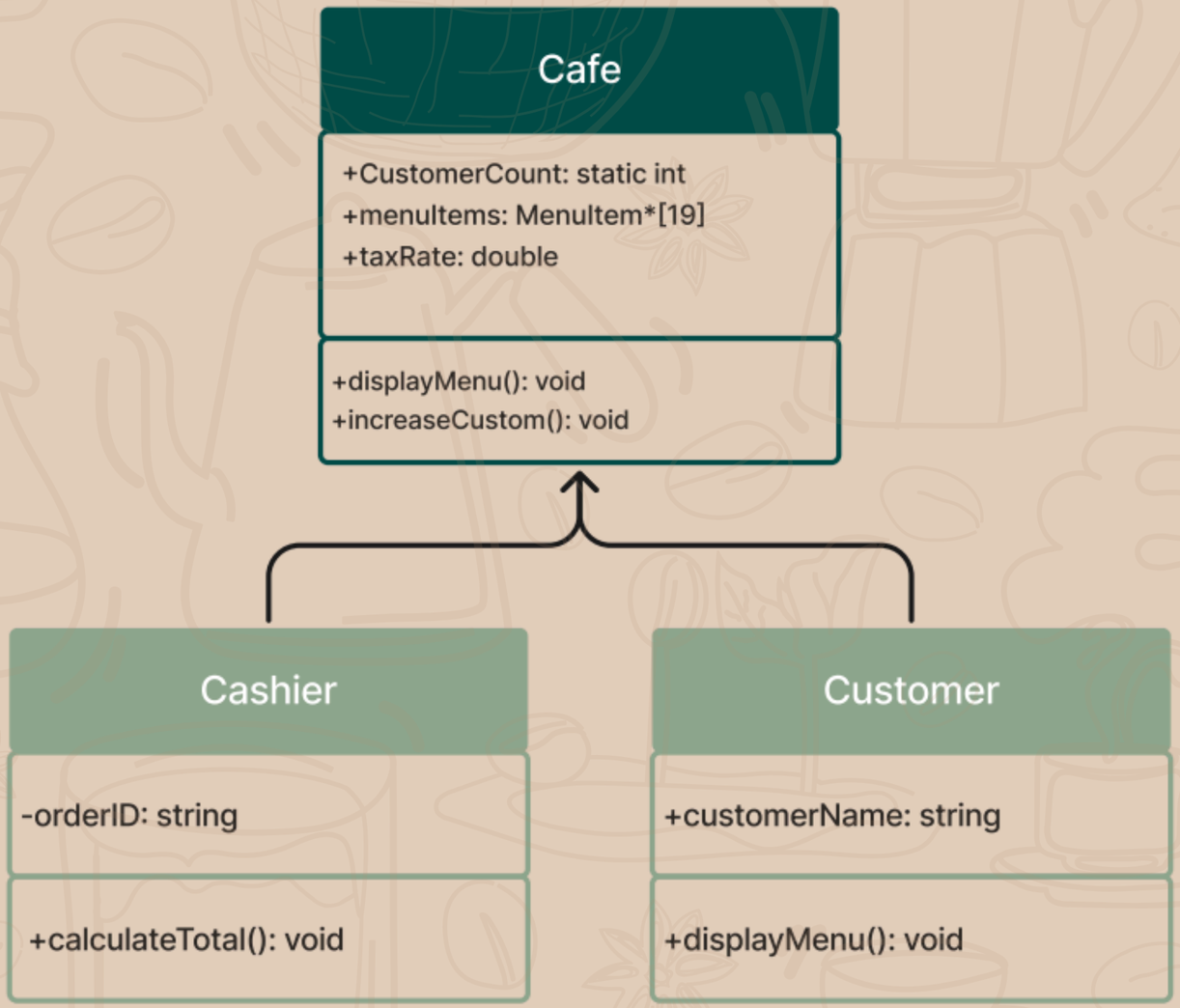
Cafe

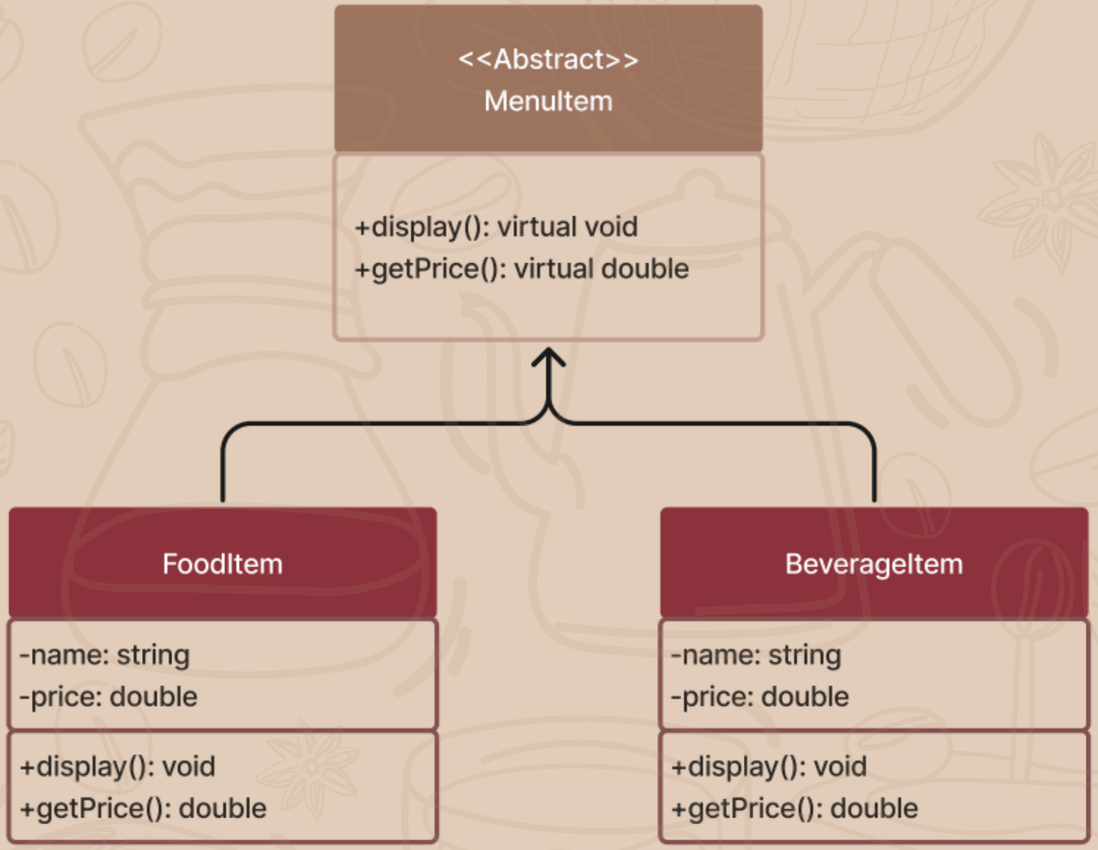
Midnight Espresso

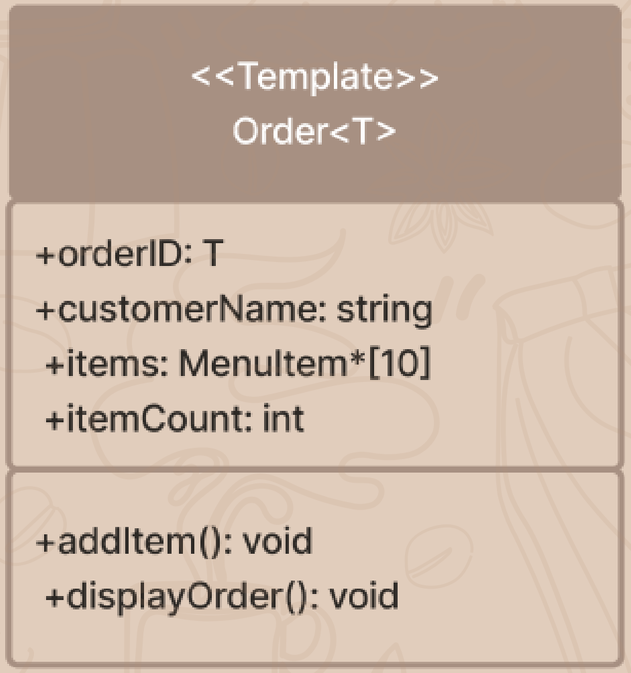
PRESENTED BY:
RANIYA YAQUB | SP23-BSI-042
TOOBA MIR | SP23-BSI-051

UML DIAGRAMS



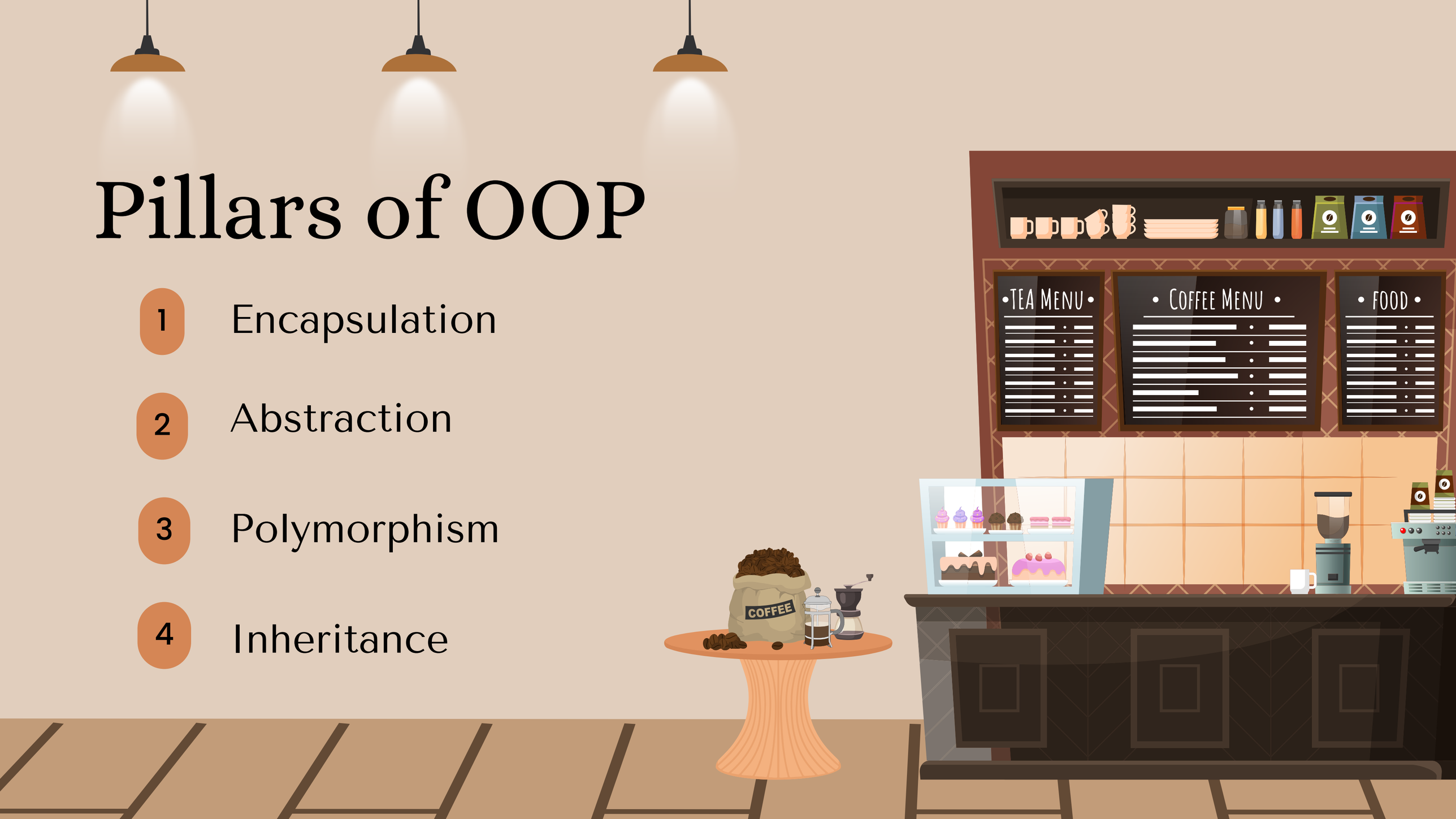
UML DIAGRAMS



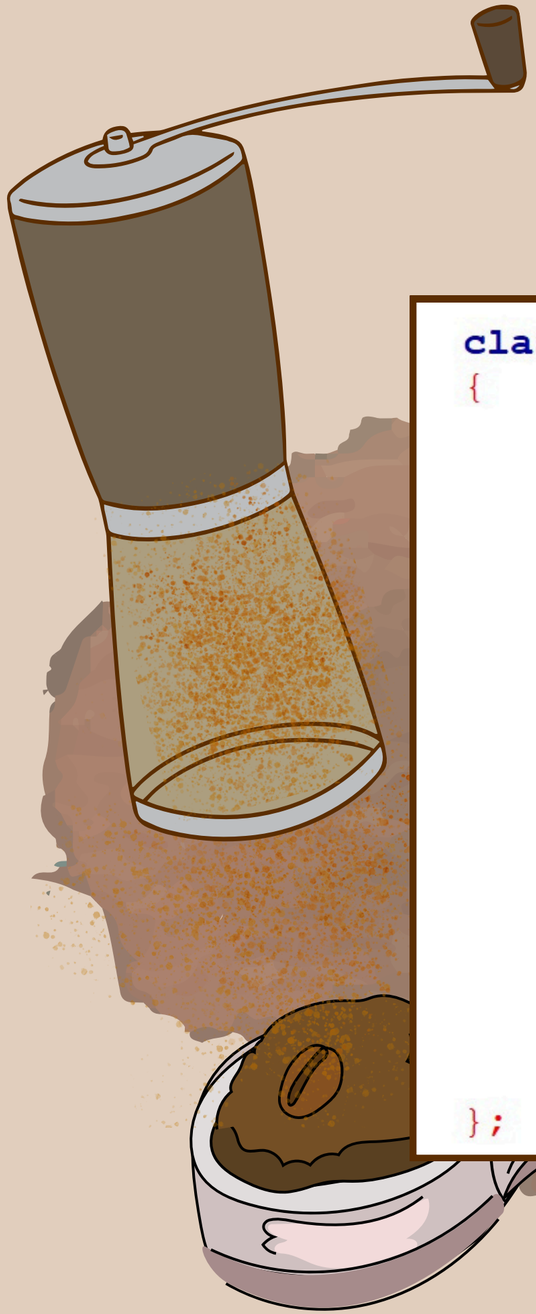


Pillars of OOP

1. Encapsulation
2. Abstraction
3. Polymorphism
4. Inheritance



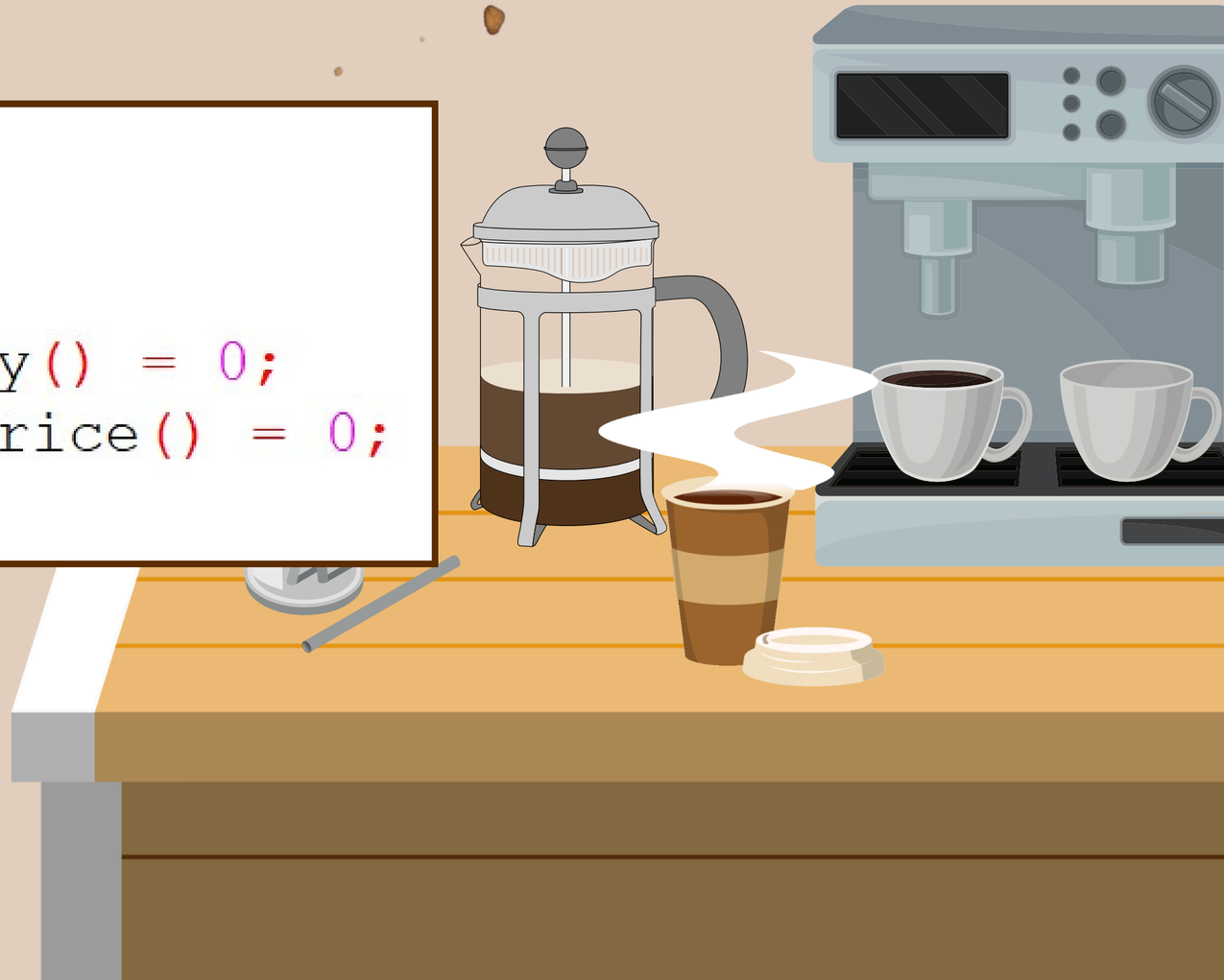
Encapsulation



```cpp
class FoodItem : public MenuItem
{
    private:
        string name;
        double price;

    public:
        FoodItem(string name, double price): name(name), price(price) {}

        void display() override
        {
            cout << name << "..................... Rs." << price << endl;
        }

        double getPrice() override
        {
            return price;
        }
};
```

Abstraction

```
class MenuItem
{
    public:
        virtual void display() = 0;
        virtual double getPrice() = 0;
};
```



Polymorphism

```cpp
class FoodItem : public MenuItem
{
    private:
        string name;
        double price;

    public:
        FoodItem(string name, double price): name(name), price(price) {}

        void display() override
        {
            cout << name << "..................... Rs." << price << endl;
        }

        double getPrice() override
        {
            return price;
        }
};
```

Inheritance

```
class Cafe
{
```

↑

```
class Cashier : public Cafe
{
```

Exception Handling

```
while (true)
{
    cout << "\nEnter the number of the item you want to order (0 to finish): ";
    cin >> itemChoice;
    cin.ignore();

    if (itemChoice == 0)
    {
        break;
    }
    else if (itemChoice > 0 && itemChoice <= 19)
    {
        customerOrder[orderItemCount++] = customer.menuItems[itemChoice - 1];
        order1.addItem(customer.menuItems[itemChoice - 1]);
        cout << "Item added to order!" << endl;
    }
    else
    {
        cout << "Invalid menu choice. Please try again." << endl;
    }
}
```

Try block

Exception thrown

```
if (Cafe::customerCount >= 5)
{
    throw "Customer limit exceeded. Cafe is now closed.";
}
```

```
catch (const char* message)
{
    cout << "\nException: " << message << endl;
    cout << "Program terminated." << endl;
}
```

Catch block

Template Class

```
template <typename T>
class Order
{
public:
    T orderID;
    string customerName;
    MenuItem* items[10];
    int itemCount;

    Order(T orderID, string customerName) : orderID(orderID), customerName(customerName), itemCount(0) {}
```

Thank You