RAMA UNIVERSITY

www.ramauniversity.ac.in

FACULTY OF ENGINEERING & TECHNOLOGY

Brajesh Mishra
Assistant Professor
Department of Computer Science & Engineering

Topics Covered

Abstraction

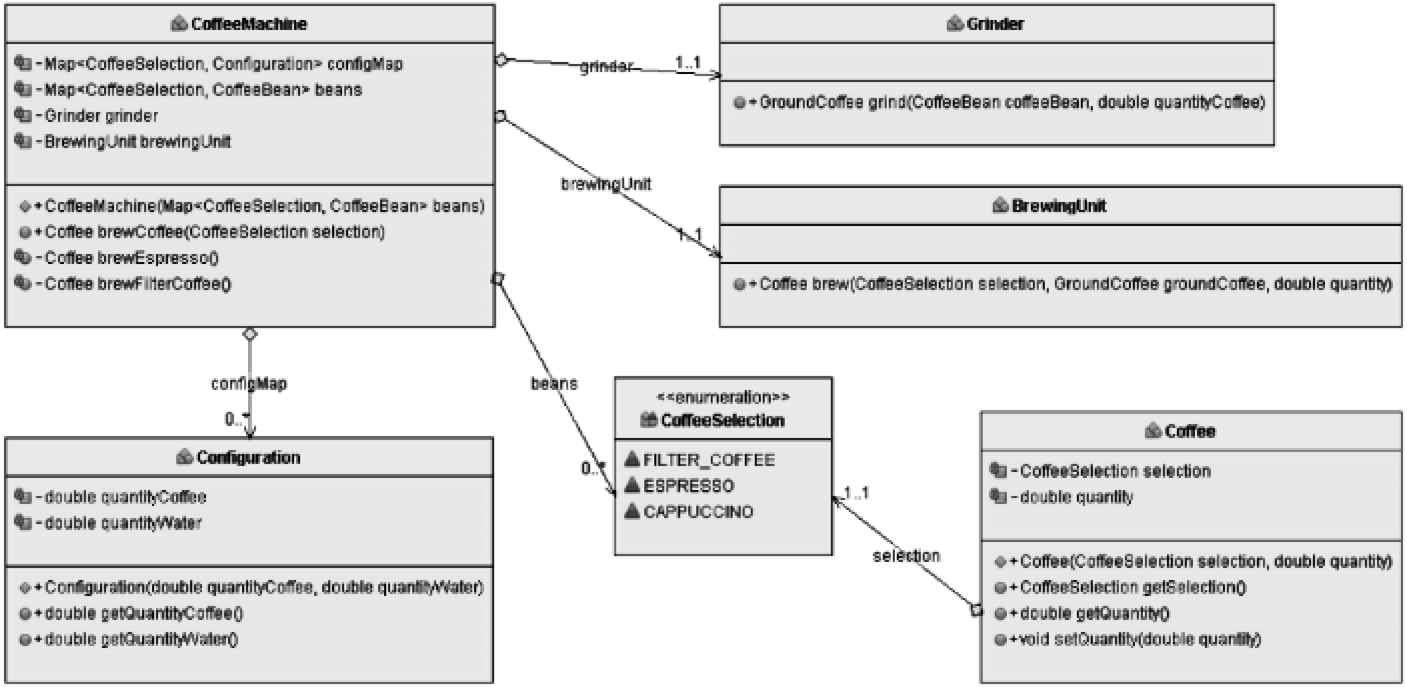
Encapsulation

Abstraction

- Abstraction is one of the key concepts of object-oriented programming (OOP) languages
- Its main goal is to handle complexity by hiding unnecessary details from the user
- It enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity
- Objects in an OOP language provide an abstraction that hides the internal implementation details

Abstraction

- As you can see in the code snippet, the constructor not only stores the provided *Map* of available *CoffeeBeans* in an internal property, it also initializes an internal *Map* that stores the configuration required to brew the different kinds of coffees and instantiates a *Grinder* and a *BrewingUnit* object.

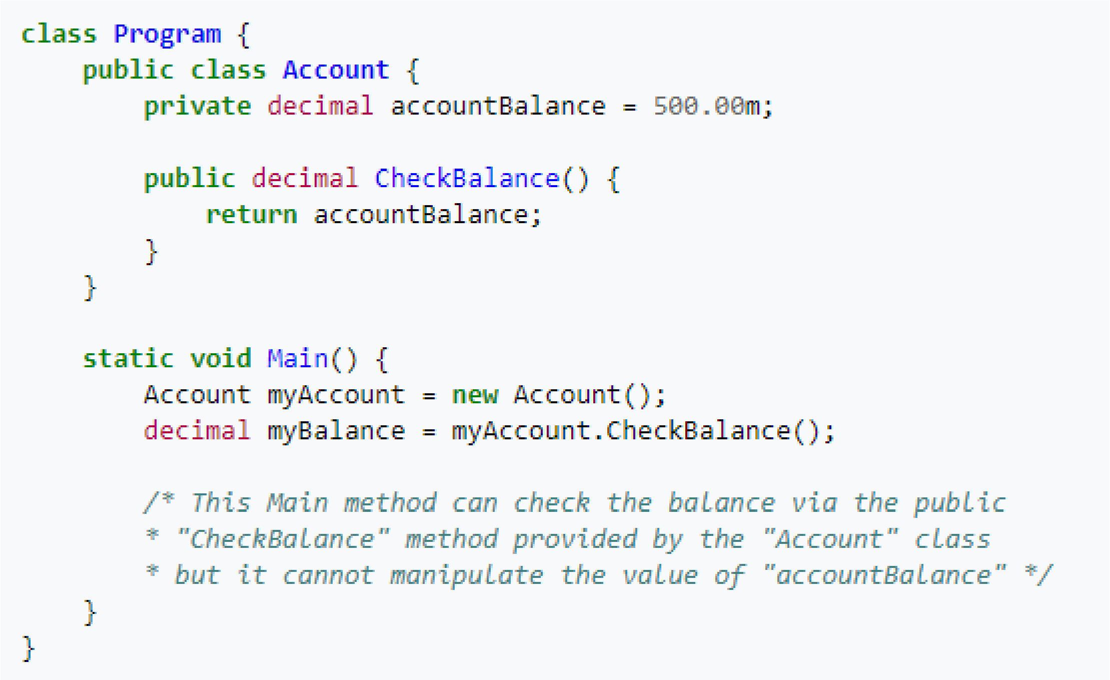


CoffeeMachine
-Map<CoffeeSelection, Configuration> configMap
-Map<CoffeeSelection, CoffeeBean> beans
-Grinder grinder
-BrewingUnit brewingUnit
+CoffeeMachine(Map<CoffeeSelection, CoffeeBean> beans)
+Coffee brewCoffee(CoffeeSelection selection)
-Coffee brewEspresso()
-Coffee brewFilterCoffee()
grinder
1..1
Grinder
+GroundCoffee grind(CoffeeBean coffeeBean, double quantityCoffee)
brewingUnit
1..1
BrewingUnit
+Coffee brew(CoffeeSelection selection, GroundCoffee groundCoffee, double quantity)
configMap
0..*
beans
0..*
<<enumeration>>
CoffeeSelection
FILTER_COFFEE
ESPRESSO
CAPPUCCINO
1..1
selection
Configuration
-double quantityCoffee
-double quantityWater
+Configuration(double quantityCoffee, double quantityWater)
+double getQuantityCoffee()
+double getQuantityWater()
Coffee
-CoffeeSelection selection
-double quantity
+Coffee(CoffeeSelection selection, double quantity)
+CoffeeSelection getSelection()
+double getQuantity()
+void setQuantity(double quantity)

Encapsulation

- Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them.

- Implementations of abstract data types, e.g., modules, offer a similar form of encapsulation



```
class Program {
    public class Account {
        private decimal accountBalance = 500.00m;

        public decimal CheckBalance() {
            return accountBalance;
        }
    }

    static void Main() {
        Account myAccount = new Account();
        decimal myBalance = myAccount.CheckBalance();

        /* This Main method can check the balance via the public
        * "CheckBalance" method provided by the "Account" class
        * but it cannot manipulate the value of "accountBalance" */
    }
}
```