Padrão de Projeto Iterator

Lucas Buarque de Araújo Barros

Problema

```
public class Manager {
    private Food[] menu;
    private ArrayList<Employee> employees;

    public Manager() {
        this.employees = new ArrayList<Employee>();
        this.menu = new Food[4];

        this.menu[0] = new Food("Lasanha", "Lasanha à Bolonhesa", 10.0);
        this.menu[1] = new Food("Pizza", "Pizza vários sabores", 20.0);
        this.menu[2] = new Food("Carpaccio", "Carpaccio de carne", 15.9);
        this.menu[3] = new Food("Espaguete", "Espaguete ao molho branco", 29.90);
    }
```

Problema

```java
public void showEmployee()
{
    for (Employee e : employees)
    {
        System.out.println(e);
    }
}
public void showMenu()
{
    for (Food f : menu)
    {
        System.out.println(f);
    }
}
```

E se existirem outras estruturas diferentes?

Consequências

- Necessidade de conhecer a representação de cada estrutura iterada.

- Necessidade de criar métodos novos para cada iteração, pois as os tipos de dados são diferentes.

- Se o sistema já estiver bastante desenvolvido, será muito difícil padronizar todos os tipos existentes para tentar obter um modelo padrão de iteração.

Iterator

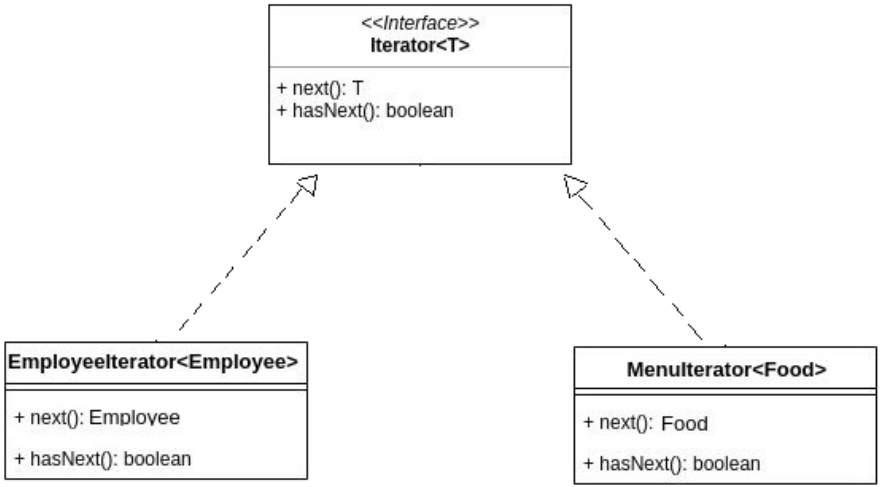
Motivação / Objetivo

- Encapsular a ideia de iteração em um objeto específico através de uma interface.

- Prover acesso à iteração de uma estrutura sem expor sua representação detalhada.

- Iteração polimórfica.

Solução


<<Interface>>
Iterator<T>
+ next(): T
+ hasNext(): boolean
EmployeeIterator<Employee>
+ next(): Employee
+ hasNext(): boolean
MenuIterator<Food>
+ next(): Food
+ hasNext(): boolean

Solução

```
public interface Iterator<T> {
    boolean hasNext();
    public T next();

}
```

Solução

```java
public class MenuIterator implements Iterator<Food> {
    private Food[] menu;
    private int index;
    public MenuIterator(Food [] menu) {
        this.menu = menu;
        this.index = 0;
    }
    @Override
    public boolean hasNext() {
        if (index >= menu.length) return false;
        return true;
    }@Override
    public Food next() {
        if (hasNext())
        {
            Food food = menu[this.index];
            this.index++;
            return food;
        }
        return null;
    }
}
```

```java
public class EmployeeIterator implements Iterator<Employee> {
    private ArrayList<Employee> employees;
    int index;
    public EmployeeIterator(ArrayList<Employee> employees) {
        this.employees = employees;
        this.index = 0;
    }
    @Override
    public boolean hasNext() {
        if (index >= this.employees.size()) return false;
        return true;
    }
    @Override
    public Employee next() {
        if (hasNext())
        {
            Employee employee = this.employees.get(index);
            this.index++;
            return employee;

        }
        return null;
    }
}
```

Solução

```
public void showInformation(Iterator it)
{
    while (it.hasNext())
    {
        System.out.println(it.next());
    }
}
```

Referências

- Use a Cabeça! Padrões de Projeto - Erick Freeman, Elisabeth Freeman.
- imasters.com.br/desenvolvimento/arquitetura-e-desenvolvimento-de-software-parte-15-iterator.
- dsc.ufcg.edu.br/~jacques/cursos/map/html/pat/iterator.htm