CAPÍTULO 11

Mapas com Lista

"Os nossos desejos são como crianças pequenas: quanto mais lhes cedemos, mais exigentes se tornam"
— Provérbio Chinês

11.1 – INTRODUÇÃO

A maneira mais simples de implementar um Mapa é guardar as associações pertencentes a ele em uma Lista.

Como sabemos quais operações queremos podemos definir o "esqueleto" da classe que vai implementar o Mapa.

```
public class MapaLista {

  private List<Associacao> associacoes = new ArrayList<Associacao>();

  public void adiciona(String placa, Carro carro) {
  }

  public Carro pega(String placa) {
  }

  public void remove(String placa) {
  }

  public boolean contemChave(String placa) {
  }
}
```

Antes de adicionar uma associação no Mapa devemos verificar se a chave da nova associação não pertence a alguma associação da Lista.

Para remover uma associação, verificar se uma chave está associada a um valor ou recuperar o valor associado a uma determinada chave é necessário percorrer a Lista.

11.2 - Operações em mapas

Vamos ver como implementar as operações nos mapas.

11.3 - Adicionar uma associação

O Mapa não pode permitir duas associações com a mesma chave. Então, fazemos uma verificação para saber se a chave já está no Mapa. Utilizamos o método `contemChave(String)` para este teste.

```
public void adiciona(String placa, Carro carro) {
  if (!this.contemChave(placa)) {
    Associacao associacao = new Associacao(placa, carro);
    this.associacoes.add(associacao);
  }
}
```

Esta operação é bem eficiente porque adicionar no fim de qualquer tipo de Lista é muito rápido.

Poderíamos ter escolhido uma outra decisão aqui: se a chave já existisse, trocamos o valor associado para este novo Carro. A API do Java trata isso desta maneira.

11.4 - Recuperar o valor associado a uma dada chave

Da maneira que foi implementado devemos percorrer todas as associações para

achar a desejada. Se a chave não estiver presente no Mapa uma exceção é lançada.

```
public Carro pega(String placa) {
  for (Associacao associacao : this.associacoes) {
    if (placa.equals(associacao.getPlaca())) {
      return associacao.getCarro();
    }
  }
  throw new IllegalArgumentException("chave não existe");
}
```

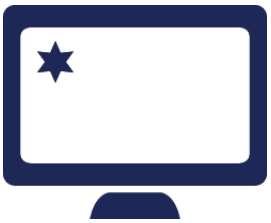
Este método também deve percorrer a Lista de associações. Então o consumo de tempo é linear.

11.5 – Remover a associação que contem uma determinada chave

Comparamos a chave recebida no parâmetro com as chaves de todas as associações da Lista. Se alguma for igual então marcamos a associação para remover. Não podemos remover dentro do **for** por causa da concorrência. Se o Mapa não tem uma associação com a chave procurada então uma exceção é lançada.

```
public void remove(String placa) {

  if (this.contemChave(placa)) {

    for (int i = 0; i < this.associacoes.size(); i++) {
      Associacao associacao = this.associacoes.get(i);

      if (placa.equals(associacao.getPlaca())) {
        this.associacoes.remove(i);
        break;
      }
    }

  } else {
    throw new IllegalArgumentException("chave não existe");
  }
}
```

O consumo de tempo deste método também é linear.



outros, com uma **assinatura** que dá acesso a todos os cursos.

Conheça os cursos online Alura.

11.6 – Verificar se uma dada chave está em alguma associação

Esta operação é simples, basta percorrer a Lista e comparar as chaves. Logo o consumo de tempo será linear.

```
public boolean contemChave(String placa) {
  for (Associacao associacao : this.associacoes) {
    if (placa.equals(associacao.getPlaca())) {
      return true;
    }
  }
  return false;
}
```

11.7 – Informar o tamanho do Mapa

Como todas as associações estão armazenadas em uma Lista, o tamanho do Mapa é o tamanho da Lista.

```
public int tamanho() {
  return this.associacoes.size();
}
```

11.8 – Exercícios: Mapas

1. Implemente a classe `Carro` no pacote **br.com.caelum.ed**.

```
public class Carro {
  private String nome;

  public Carro(String nome) {
    this.nome = nome;
  }

  public String getNome() {
    return nome;
  }

  @Override
  public String toString() {
    return "Carro: " + this.nome;
```

```
  }
}
```

2. Faça o Mapa com Listas. Assim como vimos neste capítulo.

```
public class MapaLista {

  private List<Associacao> associacoes = new ArrayList<Associacao>();

  public void adiciona(String placa, Carro carro) {
    if (!this.contemChave(placa)) {
      Associacao associacao = new Associacao(placa, carro);
      this.associacoes.add(associacao);
    }
  }

  public Carro pega(String placa) {
    for (Associacao associacao : this.associacoes) {
      if (placa.equals(associacao.getPlaca())) {
        return associacao.getCarro();
      }
    }
    throw new IllegalArgumentException("chave não existe");
  }

  public void remove(String placa) {
    if (this.contemChave(placa)) {

      for (int i = 0; i < this.associacoes.size(); i++) {
        Associacao associacao = this.associacoes.get(i);

        if (placa.equals(associacao.getPlaca())) {
          this.associacoes.remove(i);
          break;
        }
      }

    } else {
      throw new IllegalArgumentException("chave não existe");
    }
  }

  public boolean contemChave(String placa) {
    for (Associacao associacao : this.associacoes) {
      if (placa.equals(associacao.getPlaca())) {
        return true;
      }
    }
    return false;
  }
}
```

3. Faça um teste para medir a performance do nosso Mapa.

```
public class TesteTempoMapaLista {

  public static void main(String[] args) {
```

```
    MapaLista mapaLista = new MapaLista();
    int numeroDeElementos = 15000;
    long inicio = System.currentTimeMillis();

    for (int i = 0; i < numeroDeElementos; i++) {
      mapaLista.adiciona("" + i, new Carro("c" + i));
    }

    for (int i = 0; i < numeroDeElementos; i++) {
      mapaLista.pega("" + i);
    }

    for (int i = 0; i < numeroDeElementos; i++) {
      mapaLista.contemChave("" + i);
    }

    for (int i = 0; i < numeroDeElementos; i++) {
      mapaLista.remove("" + i);
    }


    long fim = System.currentTimeMillis();

    System.out.println("Tempo: " + (fim - inicio)/1000.0);
  }
}
```


CAPÍTULO ANTERIOR:

Armazenamento Associativo

PRÓXIMO CAPÍTULO:

Mapas com Espalhamento

Você encontra a Caelum também em:

Blog Caelum

Cursos Online

Facebook

Newsletter

Casa do Código

Twitter