POO - PROGRAMAÇÃO ORIENTADA A OBJETOS

HashMap
Trabalhando com Listas key-value

Rodrigo R Silva

Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense
Campus Bagé

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Nesta Aula Veremos...

Introdução

Contextualizando...

- HashMap trabalha com o conceito de key-value pairs.
- Cada elemento de sua lista possui uma chave e valor associado.
- Podemos realizar uma busca rápida pela chave que desejamos, sem precisar percorrer toda lista ou saber o index/posição que desejamos consultar.

HashMap

Estrutura do HashMap

- O HashMap implementa a interface Map T<K,V>, Cloneable e Serializable.
- Importante é apenas que ele implementa Map.
- A própria implementação do Map T<K,V> usa Generics para atribuir um key-value para a lista.
- Com o HashMap e Generics podemos especificamente dizer qual o tipo da nossa chave (string, int, double e etc) e o tipo do nosso valor, podendo diferir sem problema.

Capacidade Inicial

- Quando se instancia um HashMap a sua capacidade inicial é 16.
- Consegue-se inserir até 16 elementos no Map, sem a necessidade de criar novas posições.
- Caso deseje, também pode instanciar um HashMap com mais ou menos de 16 posições, fica ao critério e análise.

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Load Factor

- O atributo Load Factor está intrinsecamente ligado ao tamanho do HashMap.
- O load factor é um atributo que mensura em que momento o HashMap deve dobrar seu tamanho.
- Antes que possa preencher as 16 posições, em algum momento o tamanho do HashMap irá dobrar de 16 para 32.

Rehash

```java
/*
 * Supondo que o load factor e o initial size sejam padrões,
 * consideramos as variáveis abaixo.
 * Supor um cálculo genérico abaixo, para entender
 * o funcionamento do load factor.
 * É óbvio que este cálculo já está dentro da classe HashMap e
 * não precisa fazê-lo, apenas entendê-lo.
 * */

//Tamanho Inicial da Lista
int initialSize = 16;

//Valor do Load Factor
double loadFactor = 0.75;

/*
 * Aqui é o ponto mais importante do cálculo. Multiplicando
 * o tamanho inicial pelo load factor temos um valor que corresponde
 * ao tamanho máximo suportado pela lista.
 *
 * No nosso caso o resultado será igual a 12. Isso significa
 * que ao inserirmos 12 elementos em nosso HashMap,
 * a lista dobrará de tamanho, ou seja, terá tamanho = 32.
 *
 * Depois o load factor é calculado novamente para o
 * novo tamanho (32) e assim sucessivamente.
 * */
double sizeToRehash = initialSize * loadFactor;
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Rehash

- Não é aconselhável mudar o valor do load factor, a não ser que tenha certeza do que está fazendo.
- A cada duplicação da lista é realizado um ’Rehash’ na tabela que consequentemente requer mais processamento.

Aplicação

Exemplo 1

```
public class ExemploHashmap {

    public static void main(String[] args) {

        Map<String,String> examplo = new HashMap<String,String>();

          examplo.put( "K1", new String( "V1" ));
          examplo.put( "K2", "V2" );
          examplo.put( "K3", "V3" );
          examplo.put( "K4", "V4" );
          examplo.put( "K5", "V5" );
          examplo.put( "K6", "V6" );
          examplo.put( "K7", "V7" );
          examplo.put( "K8", "V8" );
          examplo.put( "K9", "V9" );
          examplo.put( "K10", "V10" );
          examplo.put( "K11", "V11" );
          examplo.put( "K12", "V12" );

          /* LIMITE DE INSERÇÃO - Aqui é o limite de acord com o load factor, ou seja,
           * quando o elemento 13 for inserido ocorrerá um Rehash na nossa lista.*/

           examplo.put( "K13", "V13" );
           System.out.println("Rehash ocorrendo agora ! "
                + "Nosso HashMap terá tamanho igual a 32 a partir daqui");

          examplo.put( "K14", "V14" );
          examplo.put( "K15", "V15" );
          examplo.put( "K16", "V16" );
    }
}
```

Observações

- O método put() para inserir um novo par de elementos em nossa lista.
- O “rehash” só ocorre depois da inserção do elemento 13, isso porque mesmo o limite sendo 12 não é necessário ainda realizar o “rehash”, pois pode ser que paremos por ali e o “rehash” terá sido desnecessário.
- O método containsKey() irá procurar por uma chave dentro da tabela.
- O valor da chave especificada neste método deve ser do mesmo tipo especificado em HashMap T<K,V>.

Exemplo 2

```
String chaveProcurada = "K1";


if ( examplo.containsKey( chaveProcurada ) ) {
 System.out.println("Valor da Chave "+chaveProcurada+
  " = "+examplo.get(chaveProcurada));
}else{
        System.err.println("Chave não existe");
}
```

Outros métodos de HashMap

- get() - Acessa um valor no HashMap consultando a chave. Exemplo: examplo.get("K1");
- remove() - Remove uma valor no HashMap pela chave informada. Exemplo: examplo.remove("K2");
- clear() - Remove todos os itens do HashMap. Exemplo: examplo.clear();
- keySet() - Retorna um set contendo todas as chaves do HashMap. Exemplo:

```
Set<String> chaves = exemplo.keySet();
for ( String chave : chaves ) {
    System.out.println( chave );
}
```

- size() - Retorna a quantidade de itens existentes. Exemplo: exemplo.size();
- values() - Retorna os valores contidos no HashMap. Exemplo:

```
for (String i : exemplo.values()) {
    System.out.println(i);
{
```

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Exemplo 3 - Usando foreach

```
    for (String chave : examplo.keySet()) {
        //Capturamos o valor a partir da chave
        String valor = examplo.get(chave);
        System.out.println(chave + " = " + valor);
}
```

Objetos Customizados

- É muito importante ficar atento a alguns aspectos quando deseja-se utilizar objetos customizados como valores ou chaves no seu HashMap, objetos como: Um DTO de Funcionario, PessoaFisica e assim por diante.
- Dois métodos são obrigatórios: equals() e hashCode(), caso estes não sejam implementados adequadamente o HashMap terá sérios problemas de busca e organização.

Considerações

Considerações

- Em algumas situações é importante ter noção do que está acontecendo por “trás dos panos”. Imagine uma HashMap com 300 mil elementos e load factor 0.75, em algum momento esses 300 mil elementos irão se transformar em 600 mil e exigir muito processamento, assim pode-se trabalhar com o load factor para atrasar esse “rehash” e ganhar performance.
- O uso do HashMap é muito comum quando trabalhamos com valores “nomeados”, ou seja, não nos importa a posição do item mas sim o valor da sua chave.
- Um local onde o HashMap é utilizado com muita frequência é na parametrização de métodos, ou seja, imagine que tem-se um método que pode receber um número diversificado de parâmetros cada um com nomes distintos, pode-se usar o HashMap com o conceito de chave = nome do parâmetro e valor = valor do parâmetro.

TADS Tecnologia em Análise e Desenvolvimento de Sistemas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

OBRIGADO!