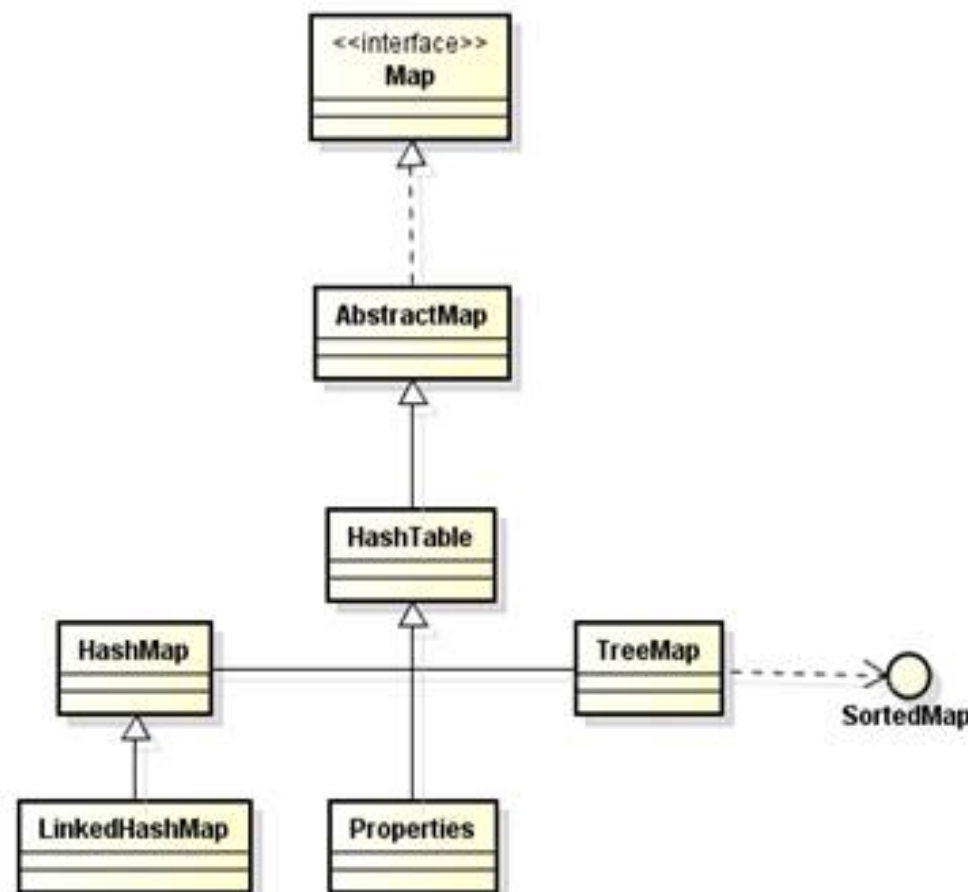


HashMap e Generics

HashMap

- A HashMap implementa a interface Map. Objetos de classes que implementam a interface Map confiam seus dados em um algoritmo de Hash (usado em Tabelas Hash). Ao lado podemos ver a hierarquia onde a HashMap está inserida.



Fonte: <http://www.linhadecodigo.com.br/artigo/3670/conhecendo-a-interface-map-do-java.aspx>

Capacidade Inicial e Fator de Carga

- A capacidade inicial é o espaço pré-alocado pela HashMap para inserir elementos. Por padrão a capacidade inicial é 16.
- O fator de carga determina quando a HashMap deve dobrar seu espaço alocado (ela não aloca no momento que se necessita, ela aloca quando atinge-se o fator de carga). Por padrão o fator de carga é 0.75, ou seja, quando utilizamos 12 elementos (16×0.75) o HashMap dobra o espaço alocado (indo para 32). Quando chegarmos a 24 elementos a HashMap irá dobrar o espaço novamente indo para 64 e assim sucessivamente.

Construtores

- Uma HashMap pode ser criada sem parâmetros, com a capacidade inicial ou com a capacidade inicial e fator de carga. Ex:

```
public class HashMap_Generics {  
    public static void main(String[] args) {  
        int capacidade_inicial = 20;    // O padrão é 16  
        float fator_carga = (float) 0.9; // O padrão é 0.75  
        Map<String, Pessoa> hash1 = new HashMap<String, Pessoa>(capacidade_inicial, fator_carga);  
        Map<String, Pessoa> hash2 = new HashMap<String, Pessoa>();  
        Map<String, Pessoa> hash3 = new HashMap<String, Pessoa>(capacidade_inicial);  
    }  
}
```

Generics

- No exemplo abaixo é possível verificar a passagem de tipos para a criação da HashMap. Isso só é possível porque a classe HashMap e a interface Map usaram o conceito de Generics.
- A vantagem de fazer isso é que não precisaremos fazer conversões (casts) ao obter elementos da HashMap.

```
public class HashMap_Generics {  
    public static void main(String[] args) {  
        int capacidade_inicial = 20;    // O padrão é 16  
        float fator_carga = (float) 0.9; // O padrão é 0.75  
        Map<String, Pessoa> hash1 = new HashMap<String, Pessoa>(capacidade_inicial, fator_carga);  
        Map<String, Pessoa> hash2 = new HashMap<String, Pessoa>();  
        Map<String, Pessoa> hash3 = new HashMap<String, Pessoa>(capacidade_inicial);  
    }  
}
```

Generics

- Generics ou “operador diamante” (<>) permite que uma classe ou interface faça referência a outras de forma genérica, tornando assim seu código mais flexível e adaptável. Vejamos parte do código da interface Map:

The diagram illustrates the generic Map interface with the following annotations:

- Tipo genérico da chave** (Green text, red arrow pointing to `K` in `Map<K extends Object, V extends Object>`)
- Tipo genérico do valor** (Green text, red arrow pointing to `V` in `Map<K extends Object, V extends Object>`)
- Retorno genérico** (Red text, dashed blue arrow pointing to `V` in `get(Object o)`)
- Parâmetro genérico** (Red text, blue arrow pointing to `K` in `put(K k, V v)`)

```
public interface Map<K extends Object, V extends Object> {  
    public static interface Entry<K extends Object, V extends Object> { ...28 linhas }  
    public int size();  
    public boolean isEmpty();  
    public boolean containsKey(Object o);  
    public boolean containsValue(Object o);  
    public V get(Object o);  
    public V put(K k, V v);  
    public V remove(Object o);  
}
```

Generics – Classe HashMap

```
public class HashMap<K extends Object, V extends Object> extends AbstractMap<K, V> implements Map<K, V>, Cloneable, Serializable {
```

K e V genéricos e herdando de Object



```
public V put(K k, V v) {  
    // compiled code  
}
```

Código compilado (inacessível)



Usando HashMap

- O primeiro método que precisamos aprender é o método put. Esse método coloca um elemento na HashMap que pode ser buscado pela chave informada.
- No nosso caso, inseriremos pessoas e a chave de cada uma será seu cpf.

```
public class HashMap_Generics {  
    public static void main(String[] args) {  
        int capacidade_inicial = 20;        // O padrão é 16  
        float fator_carga = (float) 0.9;    // O padrão é 0.75  
        Map<String, Pessoa> hash = new HashMap<String,Pessoa>(capacidade_inicial, fator_carga);  
        Scanner leitor = new Scanner(System.in);  
        // Lendo as Pessoas e colocando na HashMap (a chave será o cpf)  
        for(int i=0; i<3; i++){  
            System.out.println("Entre com o CPF");  
            String cpf = leitor.next();  
            System.out.println("Entre com o nome");  
            String nome = leitor.next();  
            Pessoa p = new Pessoa(cpf, nome);  
            hash.put(cpf, p);  
        }  
    }  
}
```


Usando HashMap

- Agora vamos aprender a buscar um elemento na HashMap:

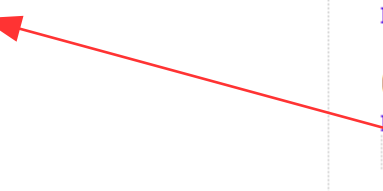
```
public class HashMap_Generics {  
    public static void main(String[] args) {  
        int capacidade_inicial = 20;        // O padrão é 16  
        float fator_carga = (float) 0.9;    // O padrão é 0.75  
        Map<String, Pessoa> hash = new HashMap<String, Pessoa>(capacidade_inicial, fator_carga);  
        Scanner leitor = new Scanner(System.in);  
        // Lendo as Pessoas e colocando na HashMap (a chave será o cpf)  
        for(int i=0; i<3; i++){  
            System.out.println("Entre com o CPF");  
            String cpf = leitor.next();  
            System.out.println("Entre com o nome");  
            String nome = leitor.next();  
            Pessoa p = new Pessoa(cpf, nome);  
            hash.put(cpf, p);  
        }  
  
        System.out.println("Digite o cpf da pessoa a ser buscada");  
        String cpf = leitor.next();  
        Pessoa pessoa;  
        // Se existe uma pessoa com esse cpf  
        if (hash.containsKey(cpf)) {  
            // Obtem a pessoa com o cpf informado  
            pessoa = hash.get(cpf);  
        }  
    }  
}
```

Usando foreach

- Podemos navegar pelos elementos da HashMap usando um Set das chaves dos elementos da HashMap. O código abaixo ilustra essa navegação:

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

```
public class Pessoa {  
    private String cpf;  
    private String nome;  
  
    public String getCpf() { ...3 linhas }  
  
    public void setCpf(String cpf) { ...3 linhas }  
  
    public String getNome() { ...3 linhas }  
  
    public void setNome(String nome) { ...3 linhas }  
  
    public Pessoa(String cpf, String nome) { ...4 linhas }  
  
    @Override  
    public String toString() {  
        return "Pessoa{" + "cpf=" + cpf + ", nome=" + nome + '}';  
    }  
}
```



Código final

```
public class HashMap_Generics {
    public static void main(String[] args) {
        int capacidade_inicial = 20; // O padrão é 16
        float fator_carga = (float) 0.9; // O padrão é 0.75
        Map<String, Pessoa> hash = new HashMap<String, Pessoa>(capacidade_inicial, fator_carga);
        Scanner leitor = new Scanner(System.in);
        // Lendo as Pessoas e colocando na HashMap (a chave será o cpf)
        for(int i=0; i<3; i++){
            System.out.println("Entre com o CPF");
            String cpf = leitor.next();
            System.out.println("Entre com o nome");
            String nome = leitor.next();
            Pessoa p = new Pessoa(cpf, nome);
            hash.put(cpf, p);
        }

        System.out.println("Digite o cpf da pessoa a ser buscada");
        String cpf = leitor.next();
        Pessoa pessoa;
        // Se existe uma pessoa com esse cpf
        if (hash.containsKey(cpf)) {
            // Obtem a pessoa com o cpf informado
            pessoa = hash.get(cpf);
        }

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

Dúvidas?

