PROGRAMAÇÃO ORIENTADA A OBJETOS

Mapas (Maps)

ROTEIRO

- **Mapas**
- **Características**
- **HashMap**
- **HashTable**
- **TreeMap**
- **Exemplos**

Mapas

- **O uso de mapas é bastante interessante quando queremos buscar um objeto, considerando que temos alguma informação sobre ele**
  - **Exemplo: Com a placa do carro ter todas as informações relevantes sobre o carro**
- **Um mapa é composto por um conjunto de associações entre um objeto chave a um objeto valor, sendo equivalente ao conceito de dicionário, presente em outras linguagens de programação**

Mapas

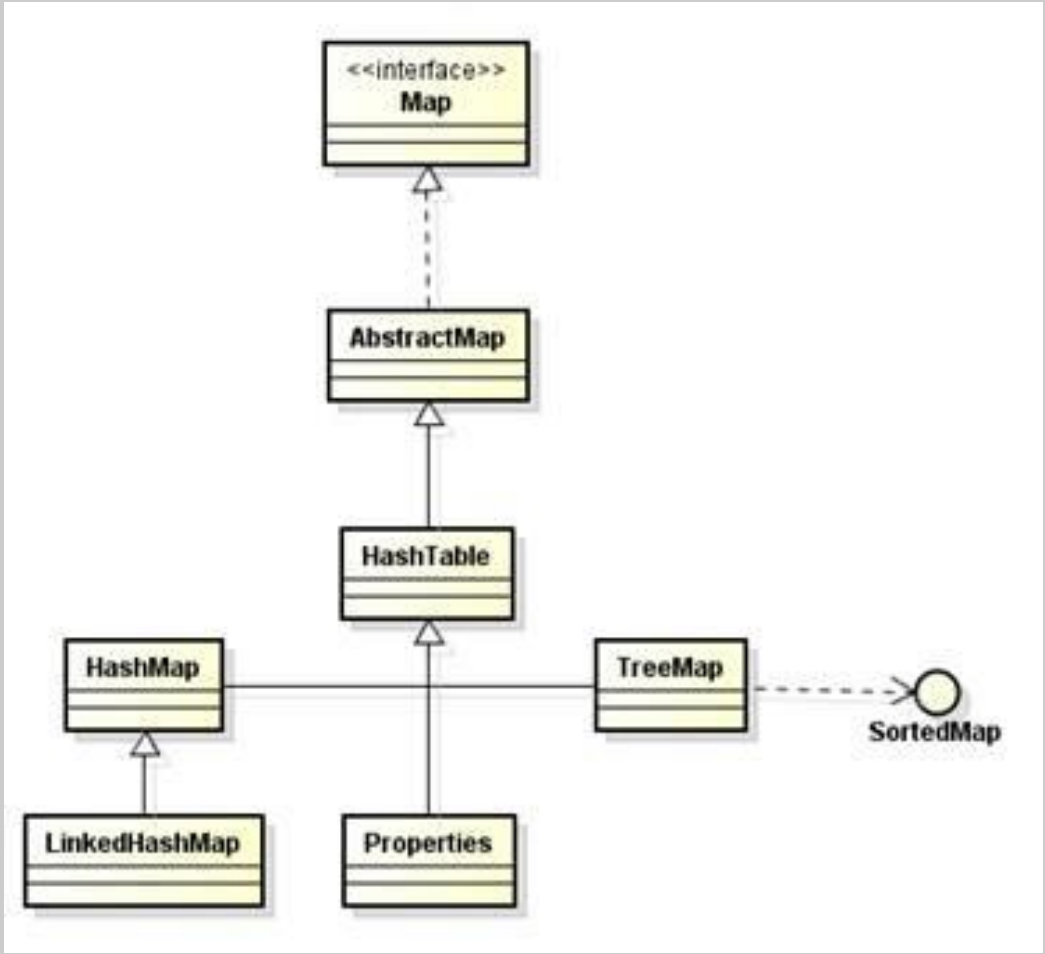
- Os objetos Map organizam seus dados em um código hash
  - O algoritmo transforma uma grande quantidade de dados em um número pequeno de informações
  - O mecanismo de busca baseia-se em construir índices
    - Exemplo: Em uma lista telefônica a letra seria o índice a ser procurado para encontrar mais facilmente o nome do titular da linha

Mapas

- Não podem conter chaves duplicadas
  - Mapeamento um para um
- Classes de Implementação
  - HashTable e HashMap: Armazena elementos na tabela hash
  - TreeMap: Armazena elementos na árvore
  - SortedMap: Estende Map mantendo suas chaves na ordem classificada
  - LinkedHashMap: Tabela hash e lista ligada
- Fazem parte do pacote java.útil
- As chaves representam objetos e os valores representam objetos associados às chaves

Mapas

- **Hierarquia da Interface Map**



Características

- **Por armazenar um par chave-valor, há dois parâmetros genéricos**

```
// Map<K, V>
Map<String, Integer> m = new HashMap<String, Integer>();
...
```

- **Algumas tarefas que podem ser feitas com Map**
  - **Inserir algo no Map**
  - **Remover algo do Map**
  - **Obter um conjunto (set) de chaves para o Map**
- **A própria implementação do Map T<K,V> usar generics para atribuir uma chave/valor**

Declaração do Map

- *Map<E> mapa = new Type();*
  - *E é o objeto declarado*
  - *Type() Tipo de objeto da coleção a ser utilizado*
- **O código abaixo testa a interface Map**

```
public class TestInterfaceMap {
  public static void main(String[] args) {
    Map<integer, string=""> mapNames = new HashMap<integer,
string="">();
    mapaNomes.put(1, "UNIVESP");
    mapaNomes.put(2, "USP");
    mapaNomes.put(3, "UNICAMP");
    mapaNomes.put(3, "UNESP");
System.out.println(mapNomes);
    //resgatando o nome da posição 2
    System.out.println(mapaNomes.get(2));
  }
}
```

HashMap

- É a implementação da interface Map
- Características
  - Os elementos não são ordenados
  - É rápida na busca e inserção de dados
  - Permite inserir valores e chaves nulas
- Métodos Importantes
  - values() - É uma Collection com todos os valores que foram associados a alguma das chaves.
  - keySet() - Retorna um Set com as chaves do mapa especificado.
  - entrySet() - Retorna um conjunto de Maps contido no mapa configurado, podendo ser possível acessar suas chaves e valores.
  - put (Key key, Value value) - Associa um valor a uma chave específica

Exemplo

- **Percorrer dados em um HashMap**

```
public class TestHashMap {
   public static void main(String args[]) {
      //Declaração do HashMap */
      HashMap<Integer, String> hm = new HashMap<Integer, String>();
      // Adição de elementos
      hm.put(1, "UNIVESP");
      hm.put(2, "USP");
      hm.put(7, "UNICAMP");
      hm.put(9, "UNESP");
      hm.put(3, "UFMG");
```

Exemplo

```
// Mostra o conteúdo, usando o Iterator
Set set = hm.entrySet();
Iterator iterator = set.iterator();
while(iterator.hasNext()) {
   Map.Entry mentry = (Map.Entry)iterator.next();
   System.out.print("A chave é: "+ mentry.getKey() + " & o Valor é:
");
   System.out.println(mentry.getValue());
}
// Obtém valores basedo na chave
String var= hm.get(2);
System.out.println("Valor no indice 2 é: "+var);
```

Exemplo

```
    // Remove valor baseado na chave
     hm.remove(3);
     System.out.println("Chave e valor do mapa após remoção:");
     Set set2 = hm.entrySet();
     Iterator iterator2 = set2.iterator();
     while(iterator2.hasNext()) {
        Map.Entry mentry2 = (Map.Entry)iterator2.next();
        System.out.print("A chave é: "+mentry2.getKey() + " & O valor é:
");
        System.out.println(mentry2.getValue());
      }
   }
}
```

HashTable

- **Estrutura de dados que utiliza hashing.**
  - **Algoritmo para determinar uma chave na tabela.**
  - **Chaves nas tabelas possuem valores associados (dados).**
  - **Cada célula na tabela é um recipiente de hash.**
    - **Lista vinculada de todos os pares chave/valor que sofrem hash para essa célula.**
    - **Minimiza colisões**

HashTable

- **Características**
  - **Usa o algoritmo hash para conversão das chaves e um mecanismo de pesquisa de valores**
  - **Apresenta uma eficiente pesquisa de elementos baseados em chave-valor**
  - **Não aceita valores nulos**
- **Declaração do HashTable**
  - ***HashTable<E> mapa = new Type<E>();***
    - ***E é o objeto declarado***
    - ***Type() Tipo de objeto da coleção a ser utilizado***

HashTable

- **Alguns métodos da classe HashTable**
  - **boolean isEmpty(): Testa se esta tabela de hash não mapeia chaves para valores.**
  - **Object remove(object key): remove a chave (e seu valor correspondente) desta tabela de hash.**
  - **int size(): Retorna o número de mapeamentos de valores-chave presentes em Hashtable.**
  - **void clear(): Remove todos os mapeamentos de valores-chave do Hashtable e o torna vazio. Limpa esta tabela de hash para que não contenha chaves.**
  - **Object clone(): Cria uma cópia superficial desta tabela de hash. Toda a estrutura da própria hashtable é copiada, mas as chaves e os valores não são clonados. Esta é uma operação relativamente cara.**

HashTable

- **Alguns métodos da classe HashTable**
  - **String toString(): Retorna a string equivalente a uma tabela hash.**
  - **boolean containsKey(Object key): Testa se o objeto especificado é uma chave nesta tabela de hash.**
  - **boolean containsValue(Object value): Testa se o objeto especificado é um valor nesta tabela de hash. Retorna verdadeiro se algum valor igual ao valor existir na tabela hash. Retorna falso se o valor não for encontrado.**
  - **Enumeration elements(): Retorna uma enumeração dos valores contidos na tabela hash.**

Exemplo com HashTable

```
public class HashtableExample {
  public static void main(String[] args) {
    Enumeration names;
   String key;
    // Cria a tabela hash
   Hashtable<String, String> hashtable = new Hashtable<String, String>();
    // Adiciona o par chave/valor
   hashtable.put("Chave1","UNIVESP");
   hashtable.put("Chave2","USP");
   hashtable.put("Chave3","UNICAMP");
   hashtable.put("Chave4","UNESP");
   hashtable.put("Chave1","Mona");
    names = hashtable.keys();
   while(names.hasMoreElements()) {
      key = (String) names.nextElement();
      System.out.println("Chave: " +key+ " & Valor: " + hashtable.get(key));
    }
  }
}
```

TreeMap

- **Características**
  - **Adição e a recuperação dos dados é igual à do HashMap.**
  - **Os dados no TreeMap são ordenados pela chave**
  - **Apenas os valores armazenados podem ser nulos, mas a chave não.**
  - **TreeMap é uma implementação baseada em árvore.**
  - **Os elementos são classificados de acordo com a ordem natural de suas chaves.**
  - **A classe TreeMap implementa a interface Map semelhante à classe HashMap.**

TreeMap

- **A principal diferença entre elas é que o HashMap é uma coleção não ordenada, enquanto o TreeMap é classificado na ordem crescente de suas chaves.**

- **Declaração do TreeMap**
  - ***TreeMap<E> mapa = new Type<E>();***
    - ***E é o objeto declarado***
    - ***Type() Tipo de objeto da coleção a ser utilizado***

Exemplo com TreeMap

- **Armazenar os mapeamentos de chave e valor no TreeMap e obter um mapeamento de chave-valor classificado ao buscar os dados do TreeMap.**

```
public class TestTreeMap {
  public static void main(String args[]) {
    //Declaração do TreeMap
    TreeMap<Integer, String> tmap;
    tmap = new TreeMap<Integer, String>();
    //Adiciona elementos
    tmap.put(1, "UNIVESP");
    tmap.put(7, "USP");
    tmap.put(6, "UNICAMP");
    tmap.put(4, "UNESP");
    tmap.put(5, "UFMG");
```

Exemplo com TreeMap

```
// Mostra o conteúdo usando o Iterator
        Set set = tmap.entrySet();
        Iterator iterator = set.iterator();
        while(iterator.hasNext()) {
           Map.Entry mentry = (Map.Entry)iterator.next();
           System.out.print("A chave é: "+ mentry.getKey() + " & O
valor é: ");
           System.out.println(mentry.getValue());
        }
    }
}
```

Referências

1. Java Como Programar: Paul Deitel & Harvey Deitel - 10ª Edição

2. Java Como Programar: Paul Deitel & Harvey Deitel - 8ª Edição

PROGRAMAÇÃO ORIENTADA A OBJETOS

Mapas (Maps)