### Funções de *hash*: Exemplo para chaves tipo `String`

```
private int hashstring(String str, int tablesize)
{
  int len=str.length();
  long hash=0;
  char[] buffer=str.toCharArray();

  int c=0;
  for (int i=0; i < len; i++)
  {
    c = buffer[i]+33;
    hash = ((hash<<3) + (hash>>28) + c);
  }

  hash = hash % tablesize;
  return (int) (hash>=0 ? hash : hash + tablesize);
}
```

- Todos os objectos em `Java` têm uma função de dispersão, `hashCode()`, que devolve um inteiro.
- Vamos utilizar esta função nas nossas tabelas de dispersão.

12.13

## 3 Factor de Carga

### Tabelas de dispersão: Factor de Carga

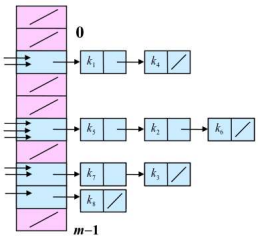
- O *factor de carga* (*load factor*) é o número de elementos na tabela dividido pelo tamanho da tabela ($\alpha = \frac{n}{m}$).
- Dimensionamento de $\alpha$:
  - um valor alto de $\alpha$ significa que vamos ter maior probabilidade de colisões;
  - um valor baixo de $\alpha$ significa que temos muito espaço desperdiçado;
  - valor recomendado para $\alpha$: entre 50% e 80%.

12.14

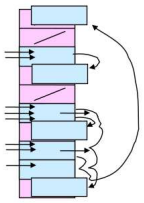
## 4 Colisões

### Resolução do Problema das Colisões

1. *Tabela de dispersão com encadeamento externo (Separate Chaining / Closed Addressing Hash Table)*
   - Múltiplos pares chaves-valor associados a um mesmo índice;
   - Cada entrada do vector contém uma lista ligada de pares chave-valor.




2. *Tabela de dispersão com encadeamento interno (Open Addressing Hash Table)*
   - No máximo, um par chave-valor em cada posição do vector;
   - No caso de colisão, segue-se um procedimento consistente para encontrar uma posição livre e armazenar aí;
   - O vector é tratado como circular.



12.15