Casos de Teste para o DOJO Hash:

Usar o método de Encadeamento Exterior, com um arquivo para tabela Hash e outro arquivo para armazenar os registros (nós da lista encadeada).

1) **testaCriaTabelaVazia**: Testa o método criaHash

Saída esperada: Arquivo de dados vazio, arquivo hash com 7 compartimentos, todos com ponteiros -1.

2) **testaBuscaChave1Tentativa**: Testa o método de busca onde a chave é encontrada na primeira tentativa.

```
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(50,  "Carlos    ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.busca(50,NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end =0 (endereço onde o cliente de chave 50 foi encontrado)

3) **testaBuscaChaveRemovida**: Testa o método de busca para uma chave que existia, mas foi removida.

```
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(50,  "Carlos    ", -1, Cliente.LIBERADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.busca(50,NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = -1 (cliente de chave 50 não encontrado)

4) **testaBuscaChave2Tentativa**: Testa o método de busca para uma chave que é encontrada na segunda tentativa.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
```

```
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria     ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(10,   "Janio     ", 5, Cliente.OCUPADO));
tabCliente.add(new Cliente(51,  "Ana       ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio  ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.busca(10,NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 3

5) **testaBuscaChaveInexistente**: Testa o método de busca para chave inexistente.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria     ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(3,   "Janio     ", 5, Cliente.OCUPADO));
tabCliente.add(new Cliente(51,  "Ana       ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio  ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.busca(10,NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = -1

6) **testaBuscaChaveReinserida**: Testa o método de busca para uma chave que existia, foi removida e depois reinserida mais ao final do arquivo.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
```

```
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria     ",  3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(10,  "Janio     ",  5, Cliente.LIBERADO));
tabCliente.add(new Cliente(51,  "Ana       ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio  ",  6, Cliente.OCUPADO));
tabCliente.add(new Cliente(10,  "Janio S.  ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.busca(10,NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 6

7) **testaInsere1Tentativa**: Testa o método de inserção.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.insere(50,"Mariana   ", NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 1 (posição onde o registro foi inserido), arquivo hash e de dados atualizado.

8) **testaInsereChaveExistente**: Testa o método de inserção para o caso de tentar inserir chave já existente.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);
```

```
int end = instance.insere(49,"Jorge     ", NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = -1 (registro não inserido, arquivos de hash e de dados não alterados)

9) **testaInsereFinalLista**: Testa o método de inserção para inserir registro no último nó da lista encadeada do compartimento ao qual ele se destina.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria    ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro    ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(3,   "Janio    ", 5, Cliente.OCUPADO));
tabCliente.add(new Cliente(51,  "Ana      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.insere(10,"Ana      ", NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 6 e arquivos de hash e de dados alterados de acordo

10) **testaInsereEspacoVazio**: Testa o método de inserção para inserir registro num nó vazio, deixado por um registro excluído.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria    ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro    ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(3,   "Janio    ", 5, Cliente.LIBERADO));
tabCliente.add(new Cliente(51,  "Ana      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.insere(10,"Ana      ", NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 3 e arquivos de hash e de dados alterados de acordo

11) **testaExclusaoChaveInexistente**: Testa o método de exclusão para excluir chave inexistente.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria     ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(3,   "Janio     ", 5, Cliente.LIBERADO));
tabCliente.add(new Cliente(51,  "Ana       ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio  ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.exclui(10, NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = -1 e arquivos de hash e de dados inalterados

12) **testaExclusaoPrimeiroNo**: Testa o método de exclusão para excluir o primeiro nó da lista encadeada de um compartimento.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria     ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(3,   "Janio     ", 5, Cliente.LIBERADO));
tabCliente.add(new Cliente(51,  "Ana       ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio  ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.exclui(59, NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 1 e arquivos de hash e de dados alterados de acordo

13) **testaExclusaoUltimoNo**: Testa o método de exclusão para excluir o último nó da lista encadeada de um compartimento.

```
tabHash.add(new CompartimentoHash(0));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(4));
tabHash.add(new CompartimentoHash(1));
tabHash.add(new CompartimentoHash(-1));
tabHash.add(new CompartimentoHash(2));
tabHash.add(new CompartimentoHash(-1));
Arquivos.salva(NOME_ARQUIVO_HASH, tabHash);
tabCliente.add(new Cliente(49,  "Joao      ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(59,  "Maria     ", 3, Cliente.OCUPADO));
tabCliente.add(new Cliente(103, "Pedro     ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(3,   "Janio     ", 5, Cliente.LIBERADO));
tabCliente.add(new Cliente(51,  "Ana       ", -1, Cliente.OCUPADO));
tabCliente.add(new Cliente(87,  "Mauricio  ", -1, Cliente.OCUPADO));
Arquivos.salva(NOME_ARQUIVO_DADOS, tabCliente);

int end = instance.exclui(87, NOME_ARQUIVO_HASH, NOME_ARQUIVO_DADOS);
```

Saída esperada: end = 5 e arquivos de hash e de dados alterados de acordo