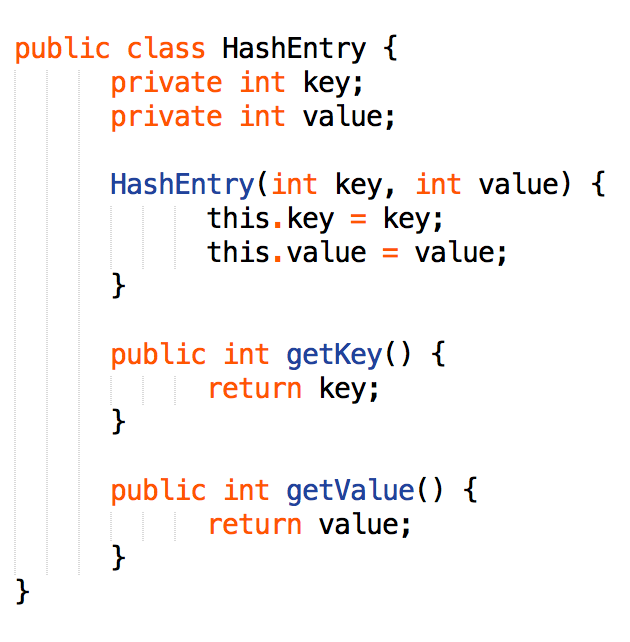
**II. Hash Table**

1. **Hashing**

* A data structure that can map keys to values with hash function
* When override equals(), also must override hashCode().
* Relationship between equals() & hashCode():
* Equals() == true -> same hashCode, but 2 different element may have same hashCode.
* So for hash table get(), we need first check with hashCode(), then equals()
* Which also means if A and B have same hash code, but A.equals(B) is false, there will be 2 key value pair in hash table.
* **Implement with Array**





* **Handle collision**
* Separate chaining
* Drawback: always allocate new nodes in the list
* Open addressing
* Addressing simply stores the colliding element in an alternate cell.
* \*Linear Probing
* \*Quadratic Probing (f(i) = i2)
* \*Double Hashing
* **Rehashing (load factor)**
* When hash table become full, insertion take longer
* Build a new table size x 2
* Extendible Hashing (used in time-sensitive applications)
* Similar to B-tree with a height = 1
* Root level is a directory whose entries point to leaves
* As the table grows, a leaf in the directory may be split to provide for this growth.
* Rehashing may be done without affecting the other leaves.

