"UML" for an **htable** class (last revised **2-10-05**)

Class: `htable`

/* a simple hash table of non-negative integers */

Member data and related details:

/* contains non-negative integers */

/* has a fixed capacity */

Constructors:

/* postcondition: creates an empty **htable** instance with a fixed capacity */

```
htable( );
```

Accessors and other constant member functions:

/* postcondition: returns **true** if htable is empty, and returns **false** otherwise */

```
bool     is_empty( ) const;
```

/* postcondition: returns **true** if htable is full (if it contains the number of items equal to its capacity), and returns **false** otherwise */

```
bool     is_full( ) const;
```

/* precondition: target >= 0 */
/* postcondition: returns **true** if <target> is in htable; returns **false** otherwise. */

```
bool     is_in_table(int target) const;
```

/* (if we were storing objects instead of simple integers, we'd likely add a member that could return a copy of a stored object given its key value...)*/

/* postcondition: returns the capacity of the htable (how many items it CAN hold) */

```
int      get_capacity( ) const;
```

/* postcondition: returns the number of elements currently in the htable */

```
int      get_size( ) const;
```

Modifiers and other modifying member functions:

/* preconditions:
 * is_full() == false
 * entry >= 0 */
/* postconditions:
 * if <entry> is already in htable, returns **false** and htable remains unchanged
 * if <entry> is not already in htable, adds it to htable and returns **true**

```
bool     insert(int entry);
```

/* preconditions: entry >= 0 */
/* postconditions:
 * if <entry> is not in htable, returns **false** and htable remains unchanged
 * if <entry> is in htable, removes it from htable and returns **true**

```
bool     remove(int entry);
```

Other member functions:

/* KLUGY --- normally would not include... BUT, for our purposes... */

/* postcondition: prints the htable's contents to the screen, separating each element with at least one blank and printing NU for never-used slots and PU for previously-used slots. */

```
void     print_guts();
```