Initial "UML" for **bag** class (last modified: 2-14-05)

adapted from Savitch and Main, "Data Structures and Other Objects Using C++"

Class: `bag`
/* an unordered collection of items of a single type, duplicates permitted */

Member data and related details:
* contains elements of type **`value_type`**
* has a size of **`size_t`**
* has a capacity, number of elements it can hold, which the class quietly expands as necessary.
* has a constant **`DEFAULT_CAPACITY`**, a **size_t.**

Constructors:
/* postcondition: creates an empty **bag** instance with an initial capacity of **DEFAULT_CAPACITY** */
```
bag();
```

/* postcondition: creates an empty **bag** instance with an initial capacity of **initial_capacity** */
```
bag(size_t initial_capacity);
```

Accessors and other constant member functions:
/* postcondition: returns the number of items in the bag. */
```
size_t      get_size()        const;
```

/* postcondition: returns the number of times that **target** is in the bag. */
```
size_t      get_count(const value_type& target) const;
```

/* postcondition: returns the current capacity of this bag (text doesn't include --- but should, if it's going to have reserve() function!) */
```
size_t      get_capacity() const;
```

Modifiers and other modifying member functions:
/* postcondition: removes all copies of **target** from the bag; returns number of copies so removed (which could be zero). */
```
size_t      erase(const value_type& target);
```

/* postconditions:
* if **target** was in the bag, then **one** copy of it has been removed; otherwise, the bag is unchanged.
* returns **true** if one copy was removed, returns **false** otherwise. */
```
bool        erase_one(const value_type& target);
```

/* postconditions: a new copy of **entry** has been inserted into the bag */
```
void        insert(const value_type& entry);
```

/* postconditions: if **new_capacity** < current bag size, bag's capacity is changed to the current bag size (will not make capacity less than the number of items already in the bag). Otherwise, the bag's current capacity is changed to **new_capacity** (even if this reduces its capacity). */
```
void        reserve(size_t new_capacity);
```