The ADT List

The ADT List

- **The ADT Bag**
  - Collection of items
  - No order to items
    - Items are in bag or not in bag
- **The ADT List**
  - Collection of items
  - Items have a position or order in the list

The ADT List

- **How a List behaves . . .**
  - adding items
  - adding items at a specified position

insert
Nachos at position **4**

	Grocery List
1	Bread
2	Oranges
3	Cheese
4	Nachoses
5	Tomatoes
6	
7	
8	
9	
10	

Pearson

THE ADT LIST

- **How a List behaves . . .**
  - adding items
  - adding items at a specified position
  - remove an item at a specific location in the list
  - replace an item in the list
    - remove, then insert
  - look at the entry at a specified position
  - count the number of items in the list
  - see whether the list is empty
  - remove all the items from the list (clear)

Low-Fat Cheese

replace
item at position 4 with
Low-Fat Cheese

	Grocery List
1	Bread
2	Apples
3	Oranges
4	Nachos
5	Cheese
6	Tomatoes
7	
8	
9	
10	

Pearson

The Abstract Class List Interface

- Position oriented data structure
- List positions begin at `1`
  - Can add entries at positions from `1` to `n+1`
  - Can remove entries at positions from `1` to `n`

```
template<class ItemType>
class ListInterface
{
public:
  /** Sees whether this list is empty. */
  virtual bool isEmpty() const = 0;

  /** Gets the current number of entries in this list. */
  virtual int getLength() const = 0;

  /** Inserts an item into this list at a given position. */
  virtual bool insert(int position, const ItemType& someItem) = 0;

  /** Deletes the entry at a given position from this list. */
  virtual bool remove(int position) = 0;

  /** Removes all entries from this list.  */
  virtual void clear() = 0;

 /** Gets the entry at the given position in this list. */
  virtual ItemType getEntry(int position) const = 0;

  /** Replaces the entry at the given position in this list. */
  virtual bool setEntry(int position, const ItemType& someItem) = 0;

  /** Destroys object & frees memory allocated by object. */
  virtual ~ListInterface() { }
}; // end ListInterface
```

Using The ADT List

The ADT List Characteristics

- **Items are referenced by position in list**
- **Each item has**
  - a unique predecessor
  - a unique successor

Pearson

Specifying ADT Operations

- **Define operation contract for the ADT list**
- **Do not specify:**
  - how to store the list items
  - how to perform the operations

- **How a List behaves . . .**
  - adding items
  - adding items at a specified position
  - remove an item at a specific location in the list
  - replace an item in the list
    - remove, then insert
  - look at the entry at a specified position
  - count the number of items in the list
  - see whether the list is empty
  - remove all the items from the list (clear)

Pearson

SPECIFYING ADT OPERATIONS

- **Define operation contract for the ADT list**
- **Do not specify:**
  - how to store the list items
  - how to perform the operations

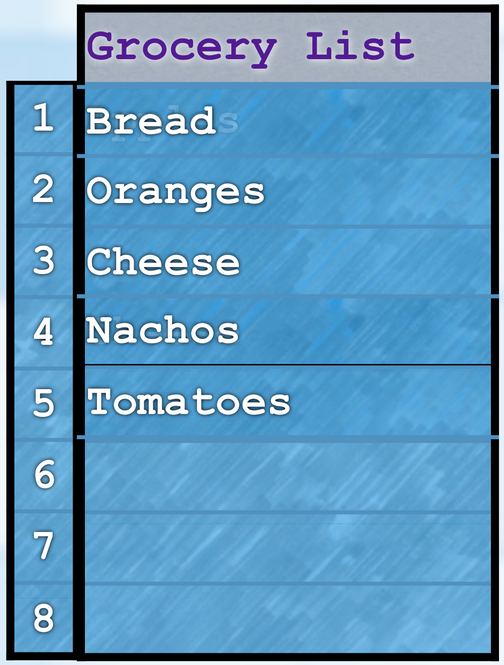
```
class ListInterface
{
public:
   virtual bool isEmpty() const = 0;
   virtual int getLength() const = 0;
   virtual bool insert(int position, const ItemType& someItem) = 0;
   virtual bool remove(int position) = 0;
   virtual void clear() = 0;
   virtual ItemType getEntry(int position) const = 0;
   virtual ~ListInterface() {  }
}; // end ListInterface
```

```
template<cla
class ListInt
{
public:
   /** Sees wh
   virtual boo

   /** Gets the
   virtual int getLength() const = 0;

   /** Inserts an entry into this list at a given position. */
   virtual bool insert(int position, const ItemType& someItem) = 0;

   /** Deletes the entry at a given position from this list. */
   virtual bool remove(int position) = 0;

   /** Removes all entries from this list.  */
   virtual void clear() = 0;

  /** Gets the entry at the given position in this list. */
   virtual ItemType getEntry(int position) const = 0;

  /** Destroys object & frees memory allocated by object. */
   virtual ~ListInterface() { }

}; // end ListInterface
```

Pearson

USING THE ADT LIST

```
class ListInterface
{
public:
  virtual bool isEmpty() const = 0;
  virtual int getLength() const = 0;
  virtual bool insert(int position, const ItemType& someItem) = 0;
  virtual bool remove(int position) = 0;
  virtual void clear() = 0;
  virtual ItemType getEntry(int position) const = 0;
  virtual ~ListInterface() { }
}; // end ListInterface
```




```
auto groceryList = std::make_shared<SomeList<std::string>>();
groceryList->insert(1, "Apples");
groceryList->insert(2, "Oranges");
groceryList->insert(3, "Cheese");
groceryList->insert(4, "Tomatoes");
groceryList->insert(1, "Bread");
groceryList->insert(4, "Nachos");
```

Pearson

USING THE ADT LIST

```
The list contains 5 entries, as follows:

Bread is entry 1
Apples is entry 2
Nachos is entry 3
Low-Fat Cheese is 4
Tomatoes is 5
```

```
class ListInterface
{
public:
  virtual bool isEmpty() const = 0;
  virtual int getLength() const = 0;
  virtual bool insert(int position, const ItemType& someItem) = 0;
  virtual bool remove(int position) = 0;
  virtual void clear() = 0;
  virtual ItemType getEntry(int position) const = 0;
  virtual ~ListInterface() { }
}; // end ListInterface
```

	Grocery List
1	Bread
2	Apples
3	Oranges
4	Nachos
5	Cheese
6	Tomatoes
7	
8	

```
auto groceryList = std::make_shared<SomeList<std::string>>();
groceryList->insert(1, "Apples");
groceryList->insert(2, "Oranges");
groceryList->insert(3, "Cheese");
groceryList->insert(4, "Tomatoes");
groceryList->insert(1, "Bread");
groceryList->insert(4, "Nachos");

groceryList->remove(4);
groceryList->setEntry(4, "Low-Fat Cheese");

int numberOfEntries = groceryList->getLength();
std::cout << "The list contains " << numberOfEntries;
std::cout << " entries, as follows:" << std::endl;
for (int position = 1; position <= numberOfEntries; position++)
   std::cout << list->getEntry(position) <<" is entry "<<position<<std::endl;
```

```
Low-Fat Cheese
```

Pearson