Designing An ADT

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

Abstract Data Type

- **Abstract Data Type (ADT)**
  - Specification for a group of values and operations on those values
- **Data Structure**
  - Implementation of an ADT within a programming language
- **Collection**
  - Object that groups other objects together
  - Provides various services to clients
    - add
    - remove
    - query

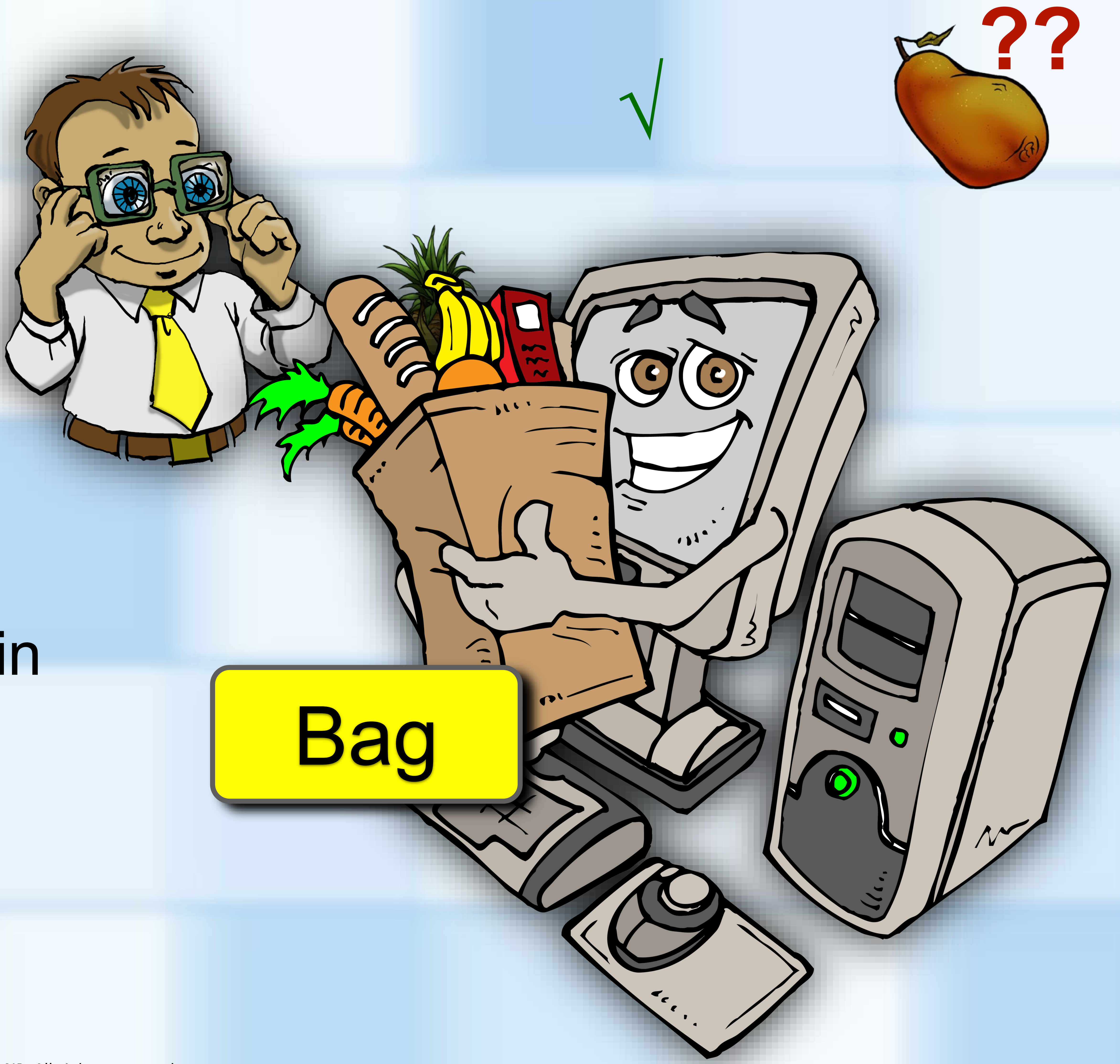
??

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

A Bag's Behaviors

- Get the number of items currently in the bag
- See whether the bag is empty
- Add a given object to the bag
- Remove an occurrence of a particular object from the bag, if possible
- Remove all objects from the bag
  - (empty or clear the bag)
- Count the number of times an object occurs in the bag
- Test whether the bag contains a particular object
- Look at all objects in the bag



Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

A Bag's Behaviors

- Get the number of items currently in the bag
- See whether the bag is empty
- Add a given object to the bag
- Remove an occurrence of a particular object from the bag, if possible
- Remove all objects from the bag
  - (empty or clear the bag)
- Count the number of times an object occurs in the bag
- Test whether the bag contains a particular object
- Look at all objects in the bag

CRC CARD

Bag

Responsibilities

Get the number of items currently in the bag
See whether the bag is empty
Add a given object to the bag
Remove an occurrence of a specific object from the bag, if possible
Remove all objects from the bag
Count the number of times a certain object occurs in the bag
Test whether the bag contains a particular object
Look at all objects that are in the bag

Collaborations

The class of objects that the bag can contain

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

A Bag's Behaviors

CRC CARD

Bag

Responsibilities

- *Get the number of items currently in the bag*
- *See whether the bag is empty*
- *Add a given object to the bag*
- *Remove an occurrence of a specific object from the bag, if possible*
- *Remove all objects from the bag*
- *Count the number of times a certain object occurs in the bag*
- *Test whether the bag contains a particular object*
- *Look at all objects that are in the bag*

Collaborations

- *The class of objects that the bag can contain*

UML NOTATION

```
Bag

+getCurrentSize(): integer
+isEmpty(): boolean
+add(someItem: T): boolean
+remove(someItem: T): boolean
+clear(): void
+getFrequencyOf(target: T): integer
+contains(target: T): boolean
+toVector(): vector<T>
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

An Interface for the ADT Bag

BagInterface.h

UML NOTATION

```
Bag
-----------------------------------
+getCurrentSize(): integer
+isEmpty(): boolean
+add(someItem: T): boolean
+remove(someItem: T): boolean
+clear(): void
+getFrequencyOf(target: T): integer
+contains(target: T): boolean
+toVector(): vector<T>
```

```cpp
#ifndef _BagInterface_h
#define _BagInterface_h

#include <vector>

template<class ItemType>
class BagInterface
{
public:
   /** Gets the current number of entries in this bag.
    @return the integer number of entries currently in the bag */
   virtual int getCurrentSize() const = 0;

   /** Sees whether this bag is empty.
    @return true if the bag is empty, or false if not */
   virtual bool isEmpty() const = 0;

   /** Adds a new entry to this bag.
    @post if successful, someItem in stored in bag and
        count of items in the bag is increased by 1
    @param someItem  the object to be added as a new entry
    @return true if addition is successful, or false if not */
   virtual bool add(const ItemType& someItem) = 0;
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

An Interface for the ADT Bag

BagInterface.h

UML NOTATION

```
                    Bag
-------------------------------------------

-------------------------------------------
+getCurrentSize(): integer
+isEmpty(): boolean
+add(someItem: T): boolean
+remove(someItem: T): boolean
+clear(): void
+getFrequencyOf(target: T): integer
+contains(target: T): boolean
+toVector(): vector<T>
```

```
/** Removes one occurrence of a given entry from this bag,
      if possible.
    @post if successful, target has been removed from the bag
        and the count of items in the bag has decreased by 1
    @param target  the entry to be removed
    @return true if removal was successful, or false if not */
   virtual bool remove(const ItemType& target) = 0;

   /** Removes all entries from this bag.
     @post bag contains no items and the count of items is 0 */
   virtual void clear() = 0;

   /** Counts the number of times a given entry appears in bag.
    @param target  the entry to be counted
    @return the number of times anEntry appears in the bag */
   virtual int getFrequencyOf(const ItemType& target) const = 0;

   /** Tests whether this bag contains a given entry.
    @param target  the entry to locate
    @return true if bag contains target, or false otherwise */
   virtual bool contains(const ItemType& target) const = 0;

   /** Returns vector with copies of all entries in the bag.
    @param bagContents  a vector
    @post bagContents contains copies of all entries in the bag */
   virtual vector<ItemType> toVector() const = 0;
};  // end BagInterface

#endif
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

USING ABSTRACT CLASS BagInterface

Implementing the interface:

```
#include " BagInterface.h"
template<class ItemType>
class Bag : public
BagInterface<ItemType>
{
 // Implementation goes here
}
```

Instantiating an object of class Bag:

```
Bag<string> shoppingList;
Bag<string> shoppingList = Bag<string>();
BagInterface<string>* shoppingList = new Bag<string>();
```

Pearson

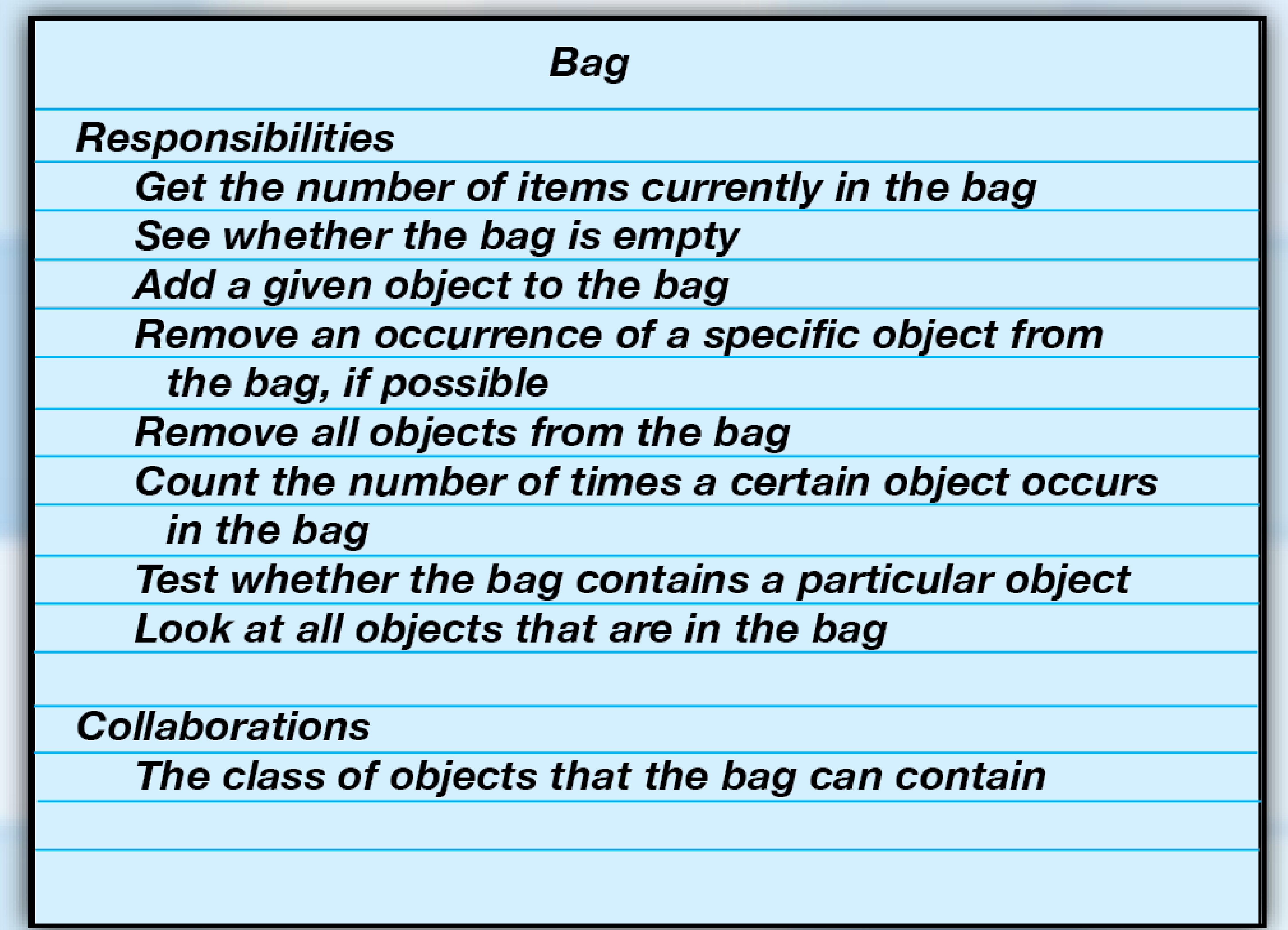
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

The ADT Bag Interface

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

A Bag's Behaviors

CRC CARD



Bag

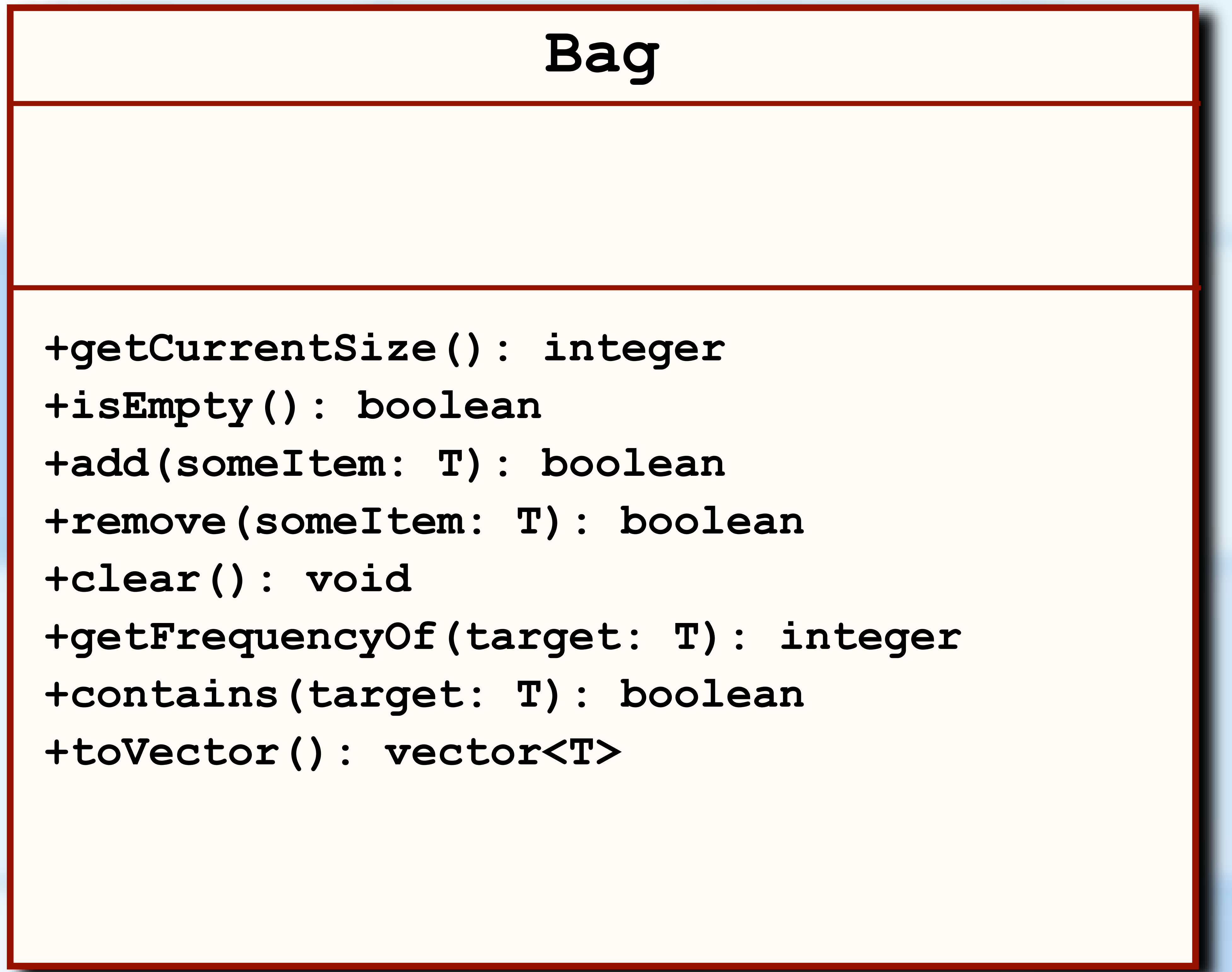
Responsibilities

Get the number of items currently in the bag
See whether the bag is empty
Add a given object to the bag
Remove an occurrence of a specific object from the bag, if possible
Remove all objects from the bag
Count the number of times a certain object occurs in the bag
Test whether the bag contains a particular object
Look at all objects that are in the bag

Collaborations

The class of objects that the bag can contain

UML NOTATION



```
Bag

+getCurrentSize(): integer
+isEmpty(): boolean
+add(someItem: T): boolean
+remove(someItem: T): boolean
+clear(): void
+getFrequencyOf(target: T): integer
+contains(target: T): boolean
+toVector(): vector<T>
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

AN INTERFACE FOR THE ADT BAG

UML NOTATION

```
Bag

+getCurrentSize(): integer
+isEmpty(): boolean
+add(someItem: T): boolean
+remove(someItem: T): boolean
+clear(): void
+getFrequencyOf(target: T): integer
+contains(target: T): boolean
+toVector(): vector<T>
```

BagInterface.h

```
#ifndef _BagInterface_h
#define _BagInterface_h
#include <vector>

template<class ItemType>
class BagInterface
{
public:
   /** Gets the current number of entries in this bag.
    @return the integer number of entries currently in the bag */
   virtual int getCurrentSize() const = 0;

   /** Sees whether this bag is empty.
    @return true if the bag is empty, or false if not */
   virtual bool isEmpty() const = 0;

   /** Adds a new entry to this bag.
    @post if successful, someItem in stored in bag and
          count of items in the bag is increased by 1
    @param someItem  the object to be added as a new entry
    @return true if addition is successful, or false if not */
   virtual bool add(const ItemType& someItem) = 0;

   /** Removes one occurrence of a given entry from this bag,
      if possible.
    @post if successful, target has been removed from the bag
          and the count of items in the bag has decreased by 1
    @param target  the entry to be removed
    @return true if removal was successful, or false if not */
   virtual bool remove(const ItemType& target) = 0;
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

AN INTERFACE FOR THE ADT BAG

UML NOTATION

```
Bag

+getCurrentSize(): integer
+isEmpty(): boolean
+add(someItem: T): boolean
+remove(someItem: T): boolean
+clear(): void
+getFrequencyOf(target: T): integer
+contains(target: T): boolean
+toVector(): vector<T>
```

BagInterface.h

```
   /** Removes all entries from this bag.
     @post bag contains no items and the count of items is 0 */
   virtual void clear() = 0;

   /** Counts the number of times a given entry appears in bag.
    @param target  the entry to be counted
    @return the number of times anEntry appears in the bag */
   virtual int getFrequencyOf(const ItemType& target) const = 0;

   /** Tests whether this bag contains a given entry.
    @param target  the entry to locate
    @return true if bag contains target, or false otherwise */
   virtual bool contains(const ItemType& target) const = 0;

   /** Returns vector with copies of all entries in the bag.
    @param bagContents  a vector
    @post bagContents contains copies of all entries in the bag */
   virtual vector<ItemType> toVector() const = 0;
}; // end BagInterface

#endif
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

USING ABSTRACT CLASS BagInterface

Implementing the interface:

```
#include " BagInterface.h"
template<class ItemType>
class Bag : public BagInterface<ItemType>
{
 // Implementation goes here
}
```

Instantiating an object of class Bag:

```
Bag<string> shoppingList;
Bag<string> shoppingList = Bag<string>();
BagInterface<string>* shoppingList = new Bag<string>();
```

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved