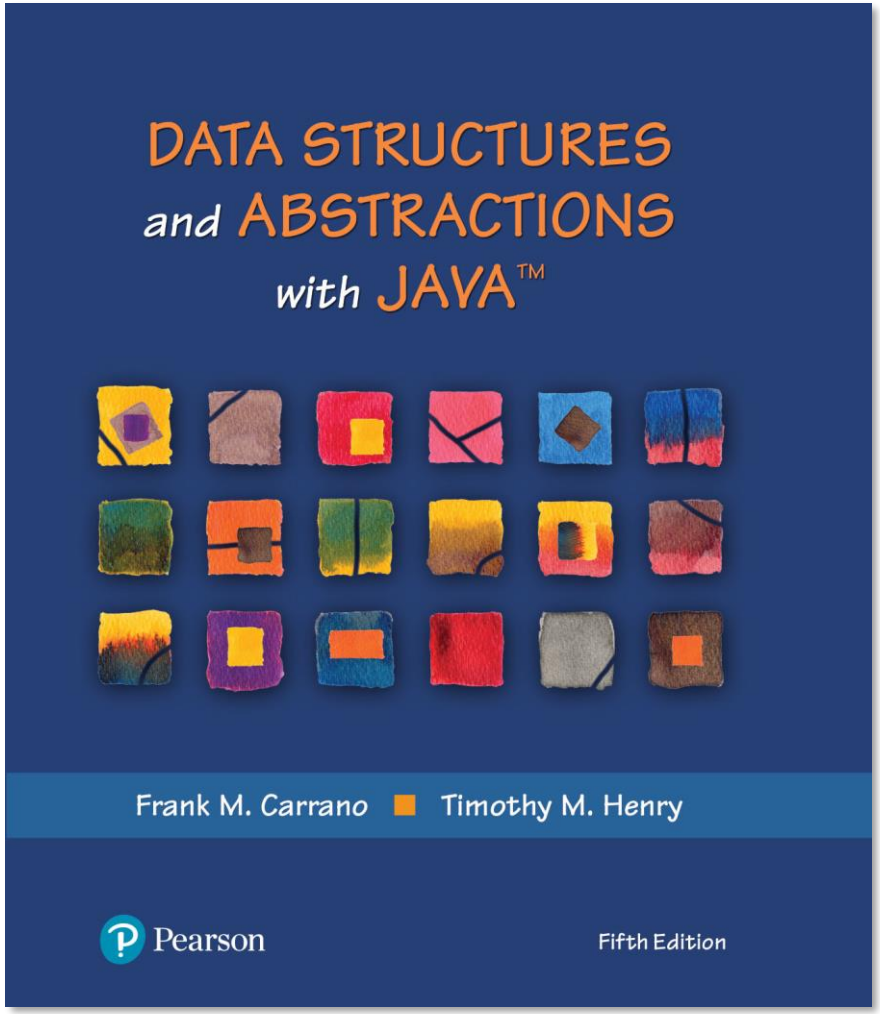
Data Structures and Abstractions with Java™

5th Edition



Chapter 1

Bags

What Is an Iterator?

- An object that traverses a collection of data
- During iteration, each data item is considered once
  - Possible to modify item as accessed
- Should implement as a distinct class that interacts with the ADT

The ADT Bag

- Definition
  - A finite collection of objects in no particular order
  - Can contain duplicate items
- Possible behaviors
  - Get number of items
  - Check for empty
  - Add, remove objects

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 unspecified object from the bag
Remove a particular 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

FIGURE 1-1 A CRC card for a class `Bag`

Specifying a Bag

- Describe its data and specify in detail the methods
- Options that we can take when add cannot complete its task:
  - Do nothing
  - Leave bag unchanged, but signal client
- Note which methods change the object or do not

Using UML Notation to Specify a Class

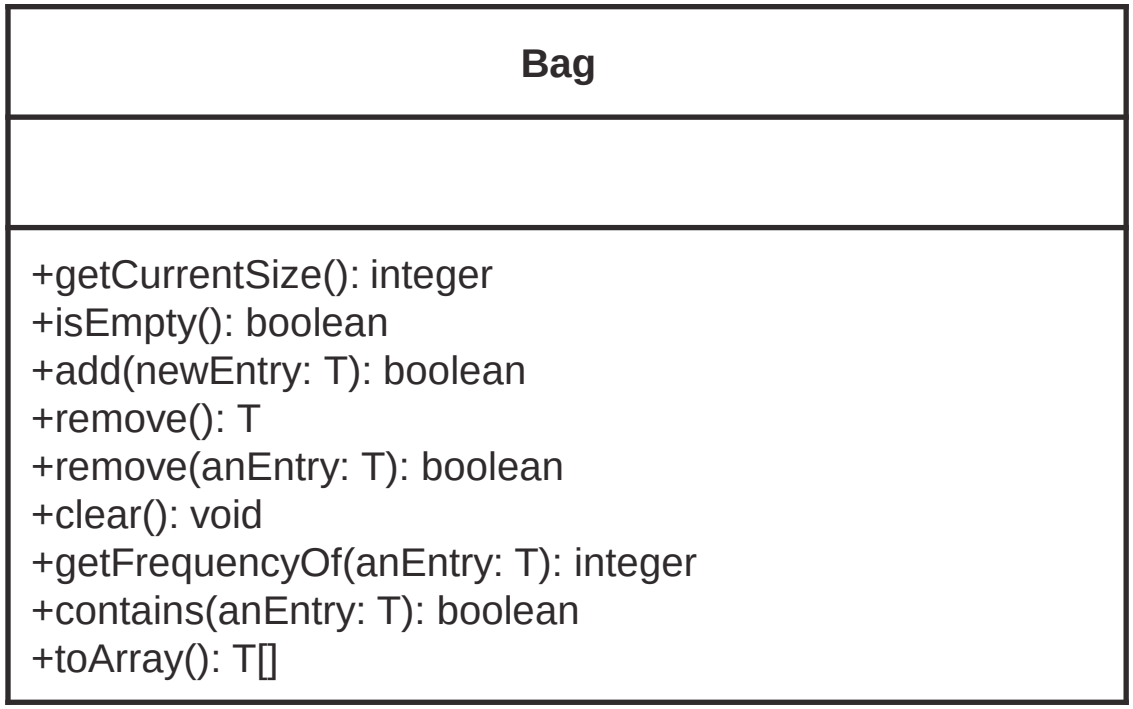


FIGURE 1-2 UML notation for the class `Bag`

Design Decision

- What to do for unusual conditions?
- Assume it won't happen
- Ignore invalid situations
- Guess at the client's intention
- Return value that signals a problem
- Return a boolean
- Throw an exception

An Interface (Part 1)

```
/** An interface that describes the operations of a bag of objects. */
public interface BagInterface<T>
{
      /** Gets the current number of entries in this bag.
       @return  The integer number of entries currently in the bag. */
      public int getCurrentSize();

      /** Sees whether this bag is empty.
       @return  True if the bag is empty, or false if not. */
      public boolean isEmpty();

      /** Adds a new entry to this bag.
          @param newEntry  The object to be added as a new entry.
          @return  True if the addition is successful, or false if not. */
      public boolean add(T newEntry);

      /** Removes one unspecified entry from this bag, if possible.
     @return  Either the removed entry, if the removal.
            was successful, or null. */
      public T remove();
```

LISTING 1-1 A Java interface for a class of bags

An Interface (Part 2)

```
    /** Removes one occurrence of a given entry from this bag, if possible.
     @param anEntry  The entry to be removed.
     @return  True if the removal was successful, or false if not. */
   public boolean remove(T anEntry);

    /** Removes all entries from this bag. */
    public void clear();

    /** Counts the number of times a given entry appears in this bag.
     @param anEntry  The entry to be counted.
     @return  The number of times anEntry appears in the bag. */
    public int getFrequencyOf(T anEntry);

    /** Tests whether this bag contains a given entry.
     @param anEntry  The entry to find.
     @return  True if the bag contains anEntry, or false if not. */
    public boolean contains(T anEntry);

    /** Retrieves all entries that are in this bag.
     @return  A newly allocated array of all the entries in the bag.
         Note: If the bag is empty, the returned array is empty. */
    public T[] toArray();
} // end BagInterface
```

LISTING 1-1 A Java interface for a class of bags

Using the ADT Bag

```
/** A class that maintains a shopping cart for an online store. */
public class OnlineShopper
{
      public static void main(String[] args)
      {
    Item[] items = {new Item("Bird feeder", 2050),
                    new Item("Squirrel guard", 1547),
                    new Item("Bird bath", 4499),
                    new Item("Sunflower seeds", 1295)};

    BagInterface<Item> shoppingCart = new Bag<>();
    int totalCost = 0;
   // Statements that add selected items to the shopping cart:
    for (int index = 0; index < items.length; index++)
    {
      Item nextItem = items[index]; // Simulate getting item from shopper
      shoppingCart.add(nextItem);
      totalCost = totalCost + nextItem.getPrice();
    } // end for

    // Simulate checkout
    while (!shoppingCart.isEmpty())
          System.out.println(shoppingCart.remove());

      System.out.println("Total cost: " + "\t$" + totalCost / 100 + "." +
                         totalCost % 100);
      } // end main
} // end OnlineShopper
```

Program Output

```
Sunflower seeds $12.95
Bird bath       $44.99
Squirrel guard      $15.47
Bird feeder         $20.50
Total cost:     $93.91
```

LISTING 1-2 A program that maintains a bag for online shopping

Example: A Piggy Bank

```
/** A class that implements a piggy bank by using a bag. */
public class PiggyBank
{
	private BagInterface<Coin> coins;

	public PiggyBank()
	{
	coins = new ArrayBag<>();
	} // end default constructor

	public boolean add(Coin aCoin)
	{
	return coins.add(aCoin);
	} // end add

	public Coin remove()
	{
	return coins.remove();
	} // end remove

	public boolean isEmpty()
	{
	return coins.isEmpty();
	} // end isEmpty
} // end PiggyBank
```

LISTING 1-3 A class of piggy banks

Example: Using A Piggy Bank (Part 1)

```
/** A class that demonstrates the class PiggyBank. */
public class PiggyBankExample
{
      public static void main(String[] args)
      {
      PiggyBank myBank = new PiggyBank();

      addCoin(new Coin(1, 2010), myBank);
      addCoin(new Coin(5, 2011), myBank);
      addCoin(new Coin(10, 2000), myBank);
      addCoin(new Coin(25, 2012), myBank);

      System.out.println("Removing all the coins:");
    int amountRemoved = 0;

    while (!myBank.isEmpty())
    {
      Coin removedCoin = myBank.remove();
      System.out.println("Removed a " + removedCoin.getCoinName() + ".");
      amountRemoved = amountRemoved + removedCoin.getValue();
    } // end while

      System.out.println("All done. Removed " + amountRemoved + " cents.");
      } // end main
```

LISTING 1-4 A demonstration of the class PiggyBank

Example: Using A Piggy Bank (Part 2)

```
    private static void addCoin(Coin aCoin, PiggyBank aBank)
    {
    if (aBank.add(aCoin))
        System.out.println("Added a " + aCoin.getCoinName() + ".");
    else
        System.out.println("Tried to add a " + aCoin.getCoinName() +
                ", but couldn't");
    } // end addCoin
} // end PiggyBankExample
```

Program Output

```
Added a PENNY.
Added a NICKEL.
Added a DIME.
Added a QUARTER.
Removing all the coins:
Removed a QUARTER.
Removed a DIME.
Removed a NICKEL.
Removed a PENNY.
All done. Removed 41 cents.
```

LISTING 1-4 A demonstration of the class PiggyBank

Observations about Vending Machines

- Can perform only tasks machine's interface presents.
- You must understand these tasks
- Cannot access the inside of the machine
- You can use the machine even though you do not know what happens inside.
- Usable even with new insides.

FIGURE 1-3
A vending machine

Observations about ADT Bag

- Can perform only tasks specific to ADT
- Must adhere to the specifications of the operations of ADT
- Cannot access data inside ADT without ADT operations
- Use the ADT, even if don’t know how data is stored
- Usable even with new implementation.

Java Class Library: The Interface `Set`

```
/** An interface that describes the operations of a set of objects. */
public interface SetInterface<T>
{
      public int getCurrentSize();
      public boolean isEmpty();

      /** Adds a new entry to this set, avoiding duplicates.
         @param newEntry  The object to be added as a new entry.
         @return  True if the addition is successful, or
           false if the item already is in the set. */
      public boolean add(T newEntry);

      /** Removes a specific entry from this set, if possible.
     @param anEntry  The entry to be removed.
     @return  True if the removal was successful, or false if not. */
      public boolean remove(T anEntry);

      public T remove();
      public void clear();
      public boolean contains(T anEntry);
      public T[] toArray();
} // end SetInterface
```

Listing 1-5 A Java interface for a class of sets

End

Chapter 1