CS 367 - Introduction to Data Structures
Tuesday, January 20, 2015

Instructor

- Jim Skrentny
- skrentny@cs.wisc.edu
- 5379 CS

Today

Collections
- Bag Intro
- Abstract Data Types
- designing the Bag ADT - Java interfaces
- using the Bag ADT

Characteristics of Good & Reusable Software
Implementing the Bag ADT using Java `Object`s
Course Topics

Next Time

Read: *Introduction*, start *Lists*
@ http://pages.cs.wisc.edu/~cs367-1/
these pages are being updated

Implementing the Bag ADT
- casting when using Object
- using Java generics for generality

List ADT
- coding the ListADT as a Java interface
- using lists via the ListADT

Collections

→ What is a *collection*?

My answer: a bunch of things?

A group of items gathered into a container.

- ITEM (data): individual member
    - simple: primitives (e.g. number, char, etc.)
    - composite: references (e.g. student)

- CONTAINER (data structure): the structure used to store the collection

→ What operations can you do on a collection? Which are the most fundamental?

Add (insert)

Remove (delete)

Look up (find, or search)

Example: Bags

Concept

A general container

- it can store any type of item
- items in the collection can be all of the same type of different types
- duplicates are okay
- a unordered container
    - there is no explicit internal order
    - add is fast
    - remove (a random item) is fast
    - search is slow

Operations

add item
remove item
check if the bag is empty

Problems

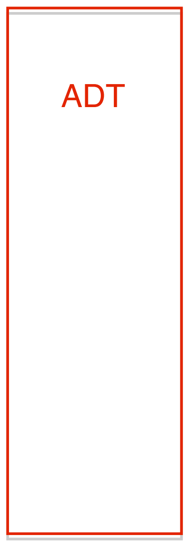
→ What problems might occur when doing Bag operations?

remove when the bag is empty
add when the bag is full

ADTs - Abstract Data Types

applications
use the ADT

there are likely to be
many if your
ADT is



implementations

the connection is
stored and how
the operations
work

list, arraylist, linked list

ADT separate applications from implementations…
ADT specifies what you can do

javadocs
- conceptual description
- list operations

java interface
    ADT are coded as interface

Designing the Bag ADT

Conceptual Description

A general unordered container storing a collection of items, where duplicates are allowed.

Public Interface

Abstract methods is public by default

```
public interface BagADT{
        void add(Object item);
        Object remove() throws NoSuchElementException;
        boolean isEmpty();
}
```

Coding Issues

remove when empty : throw exception
add when full : expand container

Example 1: Using a Bag ADT

→ **Write a code fragment**
to put the numbers 0 through 99 into a BagADT named bag.

```
BagADT bag = new ...;  //assume the bag has been instantiated for you
```

```
for ( int i = 0; i < 100; i ++){
        bag.add(i);
}
```

note: it works because add(i) is doing AUTOBOXING & UPCASTING

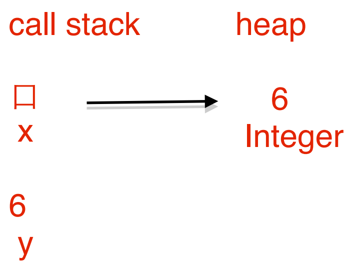
Java Autoboxing:

JAVA automatically converts between primitives and their wrapper classes

e.g.
Integer x = 6;

AUTO UNBOXING

int y = x



In the previous version of java, we need to write:
Integer x = new Integer (6);
int y = int (x)

Example 2: Using a Bag ADT

→ **Complete the printBag method**
so that it prints the contents of the parameter `bag`.
Challenge: Implement your `printBag` method so that it doesn't change the bag's contents.

```
public static void printBag(BagADT bag) {

    while ( ! bag.isEmpty())
        Object temp = bag.remove();
        S.o.pln(temp);
    }

    }
```

What makes software good?

1. works
2. easy to modify and use
3. reasonably efficient

What makes code reuseable?

1. documentation
2. modularity
    - methods
    - classes
    - interfaces
3. generality
    - object classes
    - generics

The Bag ADT and Java `Objects`

```
import java.util.*;

public interface BagADT {

    void add(Object item);
    Object remove() throws NoSuchElementException;
    boolean isEmpty();
}
```

BagADT.java

→ **Why are we using the `Object` class in our `BagADT` interface?**

For generality
- in java, an object classes reference can refer to any java object
* bags can hold any type of java object
* a single bag can hold different types

Implementing `BagADT` Using an Array of `Object` References

```
public class ArrayBag implements BagADT{

   //instance variables

        private Object [ ] items;
        private int numItems;
        private static final int INITIAL_CAPACITY = 100;



   //constructor

         public ArrayBag(){
               items = new Object [INITIAL_CAPACITY];
               numItems = 0;
         }


   //BagADT methods

            public boolean isEmpty( ) {}

            public void add(Object item) {}

            public Object remove() throws NoSuchElementException {}



   // could add other methods specific to the array implementation
}
```

Topics

Survey of Abstract Data Types (ADTs) and Data Structures (DS)

Linear -> Hierarchy -> Graph

Position-oriented -> value oriented & hybrid

Introduction to Algorithms

ADT operation
traversing
searching / sorting
hashing

Introduction to Complexity

time / space
big O notation
compare algorithms and code

Review of and Build on Java Concepts from CS 302

discussions on primitives and references
exceptions
interfaces
iterators
java collections framework

*** We assume that you are proficient at object-oriented programming in Java. If you have not learned object-oriented programming, you should complete CS 302 first. If you have learned object-oriented programming in a language like C++, you should focus time in the next two weeks to learn Java or consider taking CS 302.**