COSC 2006: Data Structures I

GTLinkedBag class

Similar to Main's LinkedBag class (3rd ED)

GTLinked bag class (1)

- This is the generic type version of the LinkedBag (Object type) class
- We also make it implement the **`Iterable`** interface which means that the new for each loop can be used.

In java.lang

```
public interface Iterable<E>
{
   public Iterator<E> iterator();
}
```

GTLinkedBag class diagram

```
GTLinkedBag<E>
---
- GTNode<E> head
- int manyNodes
---
+ GTLinkedBag( )
+ void add(E element)
+ void addAll(GTLinkedBag<E> addend)
+ GTLinkedBag<E> clone( )
+ int countOccurrences(E target)
+ E grab( )
+ GTLister<E> iterator( )
+ boolean remove(E target)
+ int size( )
+ static GTLinkedBag<E> union(GTLinkedBag<E> b1,
                              GTLinkedBag<E> b2)
+ String toString( )
```

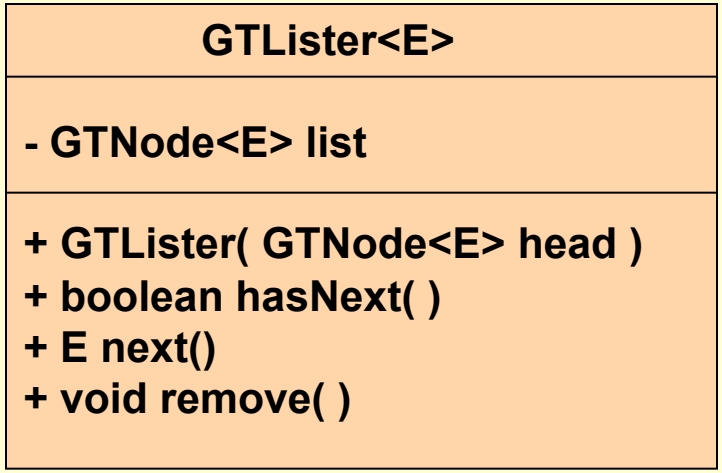
GTNode class diagram

GTNode<E>

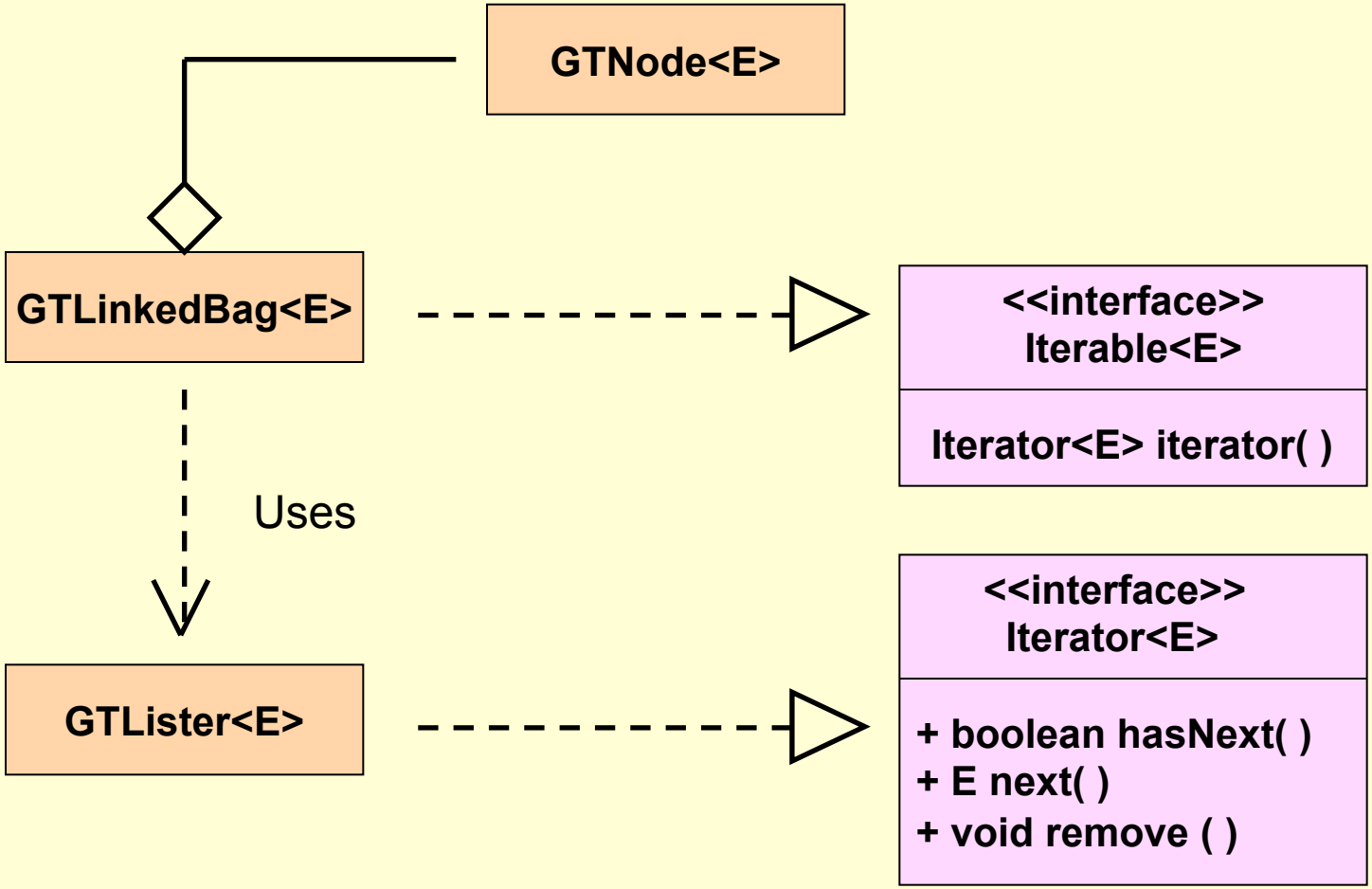
- E data
- GTNode<E> link

+ GTNode<E data, GTNode<E> link)
+ void addNodeAfter(E element)
+ void removeNodeAfter( )
+ GTNode<E> clone( )
+ E getData( )
+ GTNode<E> getLink( )
+ void setData(E data)
+ void setLink(GTNode<E> link)
+ static GTNode<E> listCopy(GTNode<E> source)
+ static Pair<E> listCopyWithTail(GTNode<E> source)
+ static int listLength(GTNode<E> head)
+ static Pair<E> listPart(GTNode<E> start, GTNode<E> end)
+ static GTNode<E> listPosition(GTNode<E> head, int position)
+ static GTNode<E> listSearch(GTNode<E> head, E target)
+ String toString( )
+ static class Pair<T>

GTLister class diagram



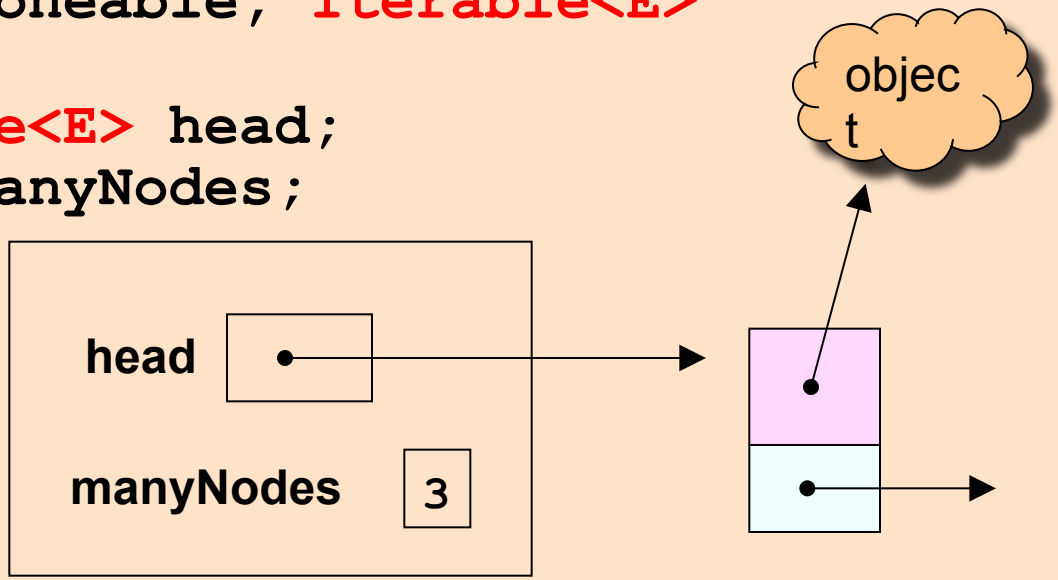
GTLinkedBag class diagram



GTLinkedBag data fields

The node type changes from Node to GTNode<E>.
We will also implement the Iterable interface so that we can use the for each loop

```
public class GTLinkedBag<E>
   implements Cloneable, Iterable<E>
{
   private GTNode<E> head;
   private int manyNodes;
```



GTLinkedBag constructor

Note: we don't have to worry about capacity so there is just one constructor with no parameters

```
public GTLinkedBag()
{
    head = null; // start with empty bag
    manyNodes = 0;
}
```

GTLinkedBag add method

Since the bag is unordered we can just add new nodes at the head of the list.
The element type is now E

```
public add(E element)
{
    head = new GTNode<E>(element, head);
}
```

GTLinkedBag addAll method

The node type is now GTNode<E> and Pair<E> is used instead of an array of two elements

```
public addAll(GTLinkedBag<E> addend)
{   if (addend == null)
        throw new IllegalArgumentException("...");
    GTNode.Pair<E> copyInfo;
    if (addend.manyNodes > 0)
    {   copyInfo = GTNode.listCopyWithTail(
                                        addend.head);
        // link addend copy to this head
        copyInfo.getTail().setLink(head);
        // set this head to head of copy
        head = copyInfo.getHead();
        manyNodes += addend.ManyNodes;
    }
}
```

GTLinkedBag clone method

The node type is now GTNode<E>

```
public GTLinkedBag<E> clone()
{
   GTLinkedBag<E> answer;
   try
   {
      answer = (GTLinkedBag<E>) super.clone();
   }
   catch (CloneNotSupportedException e)
   {
      throw new RunTimeException("...");
   }
   answer.head = GTNode.listCopy(head);
   return answer;
}
```

GTLinkedBag countOccur... (1)

The node type is now Node and parameter is of type E

```
public int countOccurrences(E type)
{
   int answer = 0;
   GTNode<E> cursor =
             GTNode.listSearch(head, target);

   while (cursor != null)
   {
      answer++;
      cursor = cursor.getLink();
      cursor = GTNode.listSearch(cursor,target);
   }
   return answer;
}
```

GTLinkedBag countOccur...(2)

Here is an alternate method that doesn't use listSearch.
Note that we need to use equals method here not ==.

```
public int countOccurrences(E target)
{
   int answer = 0;
   GTNode<E> cursor = head;
   while (cursor != null)
   {  if (target.equals(cursor.getData()))
         answer++;
      cursor = cursor.getLink();
   }
   return answer;
}
```

Note however that this version does not allow nulls in the bag

GTLinkedBag countOccur...(3)

A similar method that uses a standard for loop.
Note that we need to use equals method here not ==

```
public int countOccurrences(E target)
{
   int answer = 0;
  for (GTNode<E> cursor = head; current != null;
      cursor = cursor.getLink())
   {
      if (target.equals(current.getData()))
         answer++;
   }
   return answer;
}
```

Note however that this version does not allow nulls in the bag

GTLinkedBag iterator method

We will explain the Iterator interface later.
GTLister implements the Iterator interface and this method makes GTLinkedBag implement the Iterable<E> interface

```
public GTLister<E> iterator()
{
   return new GTLister<E>(head);
}
```

GTLinkedBag remove method

The target has type E and the node type is GTNode<E>

```
public boolean remove(E target)
{
   GTNode<E> targetNode =
                GTNode.listSearch(head,target);
   if (targetNode == null)
      return false;
   else
   {
      targetNode.setData(head.getData());
      head = head.getLink();
      manyNodes--;
      return true;
   }
}
```

GTLinkedBag size method

Return number of nodes currently in the list

```
public int size()
{
   return manyNodes;
}
```

GTLinkedBag union method

For this static method the class is now GTLinkedBag<E>

```
public static <E> GTLinkedBag<E> union(
         GTLinkedBag<E> b1, GTLinkedBag<E> b2)
{
   if (b1 == null)
      throw new IllegalArgumentException("...");
   if (b2 == null)
      throw new IllegalArgumentException("...");

   GTLinkedBag<E> answer = new GTLinkedBag<E>();
   answer.addAll(b1);
   answer.addAll(b2);
}
```

GTLinkedBag toString method

Node type is now GTNode<E>

```
public String toString()
{   StringBuffer s = new StringBuffer();
    s.append("GTLinkedBag[");
    GTNode<E> current = head;
    while (current != null)
    {
        s.append(current.getData());
        if (current.getLink() != null)
            s.append(",");
        current = current.getLink();
    }
    s.append("]");
    return s.toString();
}
```

GTLister class (1)

This class just needs to implement the Iterator<E> interface.

```
public class GTLister<E> implements Iterator<E>
{
   private GTNode<E> list;

   public GTLister(GTNode<E> head)
   {
      list = head;
   }

   // continued next slide
```

give this constructor a reference to the head of the list you want to iterate over

GTLister class (2)

This class just needs to implement the Iterator<E> interface.

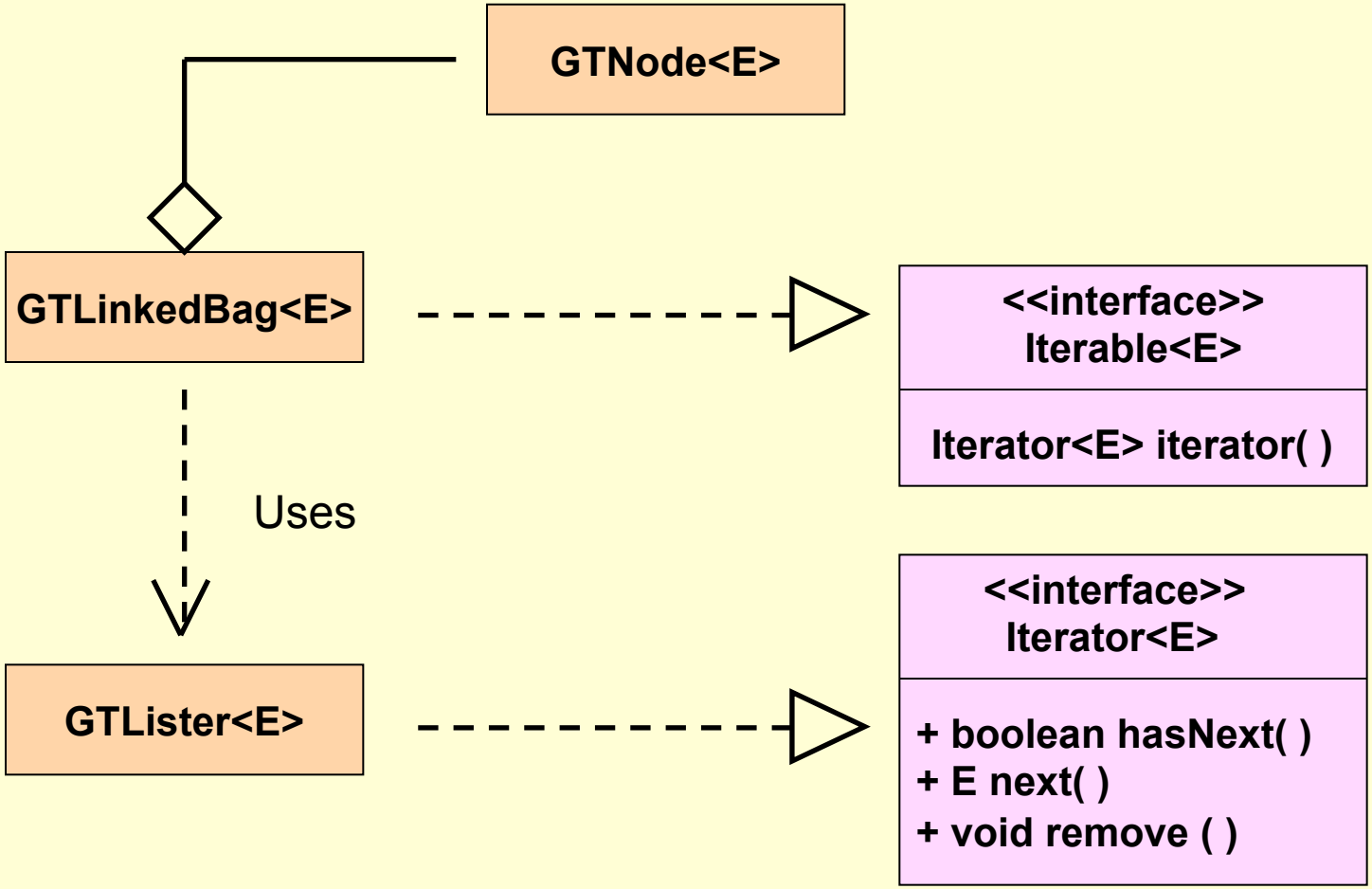
```
public boolean hasNext()
{  return list != null;
}

public E next()
{
   if (!hasNext())
      throw new NoSuchElementException("...");
   E answer = list.getData();
   list = list.getLink(); // next entry
   return answer;
}

// next slide
```

GTLister class (3)

We do not want to implement the remove method so we just throw and exception to indicate that it is not implemented.

```
    public void remove()
    {
        throw new UnsupportedOperationException(
            "Remove not implemented");
    }
}
```

GTLinkedBag class diagram



String bag example

Create a GTLinkedBag of strings, grab a few strings and display them.

```
GTLinkedBag<String> bag =
                        new GTLinkedBag<String>();
bag.add("Tom"); bag.add("Dick");
bag.add("Harry");

String s1 = bag.grab();
String s2 = bag.grab();

System.out.println("Strings grabbed are " +
   s1 + " and " + s2);
```

Integer bag example

Create a GTLinkedBag of Integer objects, using auto boxing and unboxing, grab a few integers and display them.

```
GTLinkedBag<Integer> bag =
                    new GTLinkedBag<Integer>();

bag.add(1); bag.add(2); bag.add(3);

int i1 = bag.grab();
int i2 = bag.grab();

System.out.println("Integers grabbed are " +
   i1 + " and " + i2);
```

BankAccount bag example (1)

Create a GTLinkedBag of BankAccount objects and use the bag iterator to display them

```
GTLinkedBag<BankAccount> bag =
                  new GTLinkedBag<BankAccount>();

bag.add(new BankAccount(123, "Jack", 100.0);
bag.add(new BankAccount(124, "Jill", 150.0);
bag.add(null); bag.add(null);

GTLister<BankAccount> iter = bag.iterator();
while (iter.hasNext())
{
   BankAccount b = iter.next();
   System.out.println(b);
}
```

BankAccount bag example (2)

Create a bag of BankAccount objects and display only the ones whose balance is less than 100 dollars

```
GTLinkedBag<BankAccount> bag =
                  new GTLinkedBag<BankAccount>();
bag.add(new BankAccount(123, "Jack", 100.0);
bag.add(new BankAccount(124, "Jill", 150.0);

GTLister<BankAccount> iter = bag.iterator();
while (iter.hasNext())
{
   BankAccount b = iter.next();
   if (b.getBalance() < 100.0)
      System.out.println(b);
}
```

BankAccount bag example (3)

Create a GTLinkedBag of BankAccount objects and use thenew for each loop to display them

```
GTLinkedBag<BankAccount> bag =
                  new GTLinkedBag<BankAccount>();

bag.add(new BankAccount(123, "Jack", 100.0);
bag.add(new BankAccount(124, "Jill", 150.0);
bag.add(null); bag.add(null);

for (BankAccount b : bag)
{
   System.out.println(b);
}
```