Iterators
The for loop for collections

• We already saw the special `for` loop for collections

```java
for(Item E: someCollection) { ... }
```

• This loop is implemented using an `Iterator` under the hood
  – The topic of the rest of today’s lecture
The iterator interface

• An **iterator** is an object that will return the elements of a collection, one at a time

• **interface Iterator<E>**
  – **boolean hasNext()**
    • Returns true if the iteration has more elements
  – **E next()**
    • Returns the next element in the iteration
  – **void remove()**
    • Removes from the underlying collection the last element returned by the iterator (optional operation)
Using an Iterator

• Every collection in Java implements `Iterator`
  – It is possible to get that `Iterator` and use it directly

```java
static void printAll (Collection coll) {
    Iterator iter = coll.iterator();
    while (iter.hasNext()) {
        System.out.println(iter.next());
    }
}
```

• `hasNext()` just checks if there are any more elements
• `next()` returns the next element and advances in the collection
• Note that this code is polymorphic – it will work for any collection
There is an easier way: new for loop!

- The syntax of the new statement is
  - `for(type var : array) {...} or for(type var : collection) {...}
- Example:
  ```java
  for(float x : myRealArray) {
    myRealSum += x;
  }
  ```
- For a collection class that has an Iterator, instead of
  ```java
  for (Iterator iter = c.iterator(); iter.hasNext();)
    System.out.println(iter.next( ));
  ```
you can now say
  ```java
  for (TimerTask task : c)
    System.out.println(task);
  ```
- Note that this `for` loop is implemented with an `Iterator`!
The Iterable interface

• You can also write your own classes that can be used with the new for loop
  – Must implement the `Iterable` interface

• There is just one method you need to implement
  – `public Iterator<T> iterator();`

• You don’t usually need to implement an iterator from scratch
  – Usually your class will be based on an existing collection, and you can just use that collection’s iterator
  – You can also have an `Iterator` as an instance variable
    • In this case, you need to make sure you implement the methods inside the `Iterator` interface as well

• Eclipse example: Box.java