Iterator – Look at each element in ???

- **Iterator is an interface**
  - (This concept is similar to the idea of an abstract class)
- **Something that is an Iterator must have the methods:**
  - hasNext()
  - next()
- **Look at the API**
- **Used in the following pattern:**

```java
while(???.hasNext()) {
    var = ???.next();
    //do something with var
}
```

- Don’t know the order of the elements
- Guaranteed to give you all the elements in the related set of information
Iterator

- Can’t use an iterator with an array.
  - Why?
  - What’s wrong with this picture?:

```java
int[] m = new int[n];
fillWithData(m);
while (m.hasNext()) {
    ...
```
How do we create an Iterator?

- For Collections such as ArrayLists one can get an interator using the `iterator` method:
  ```java
  // You must create iterator for ArrayList
  // Assume an ArrayList<String> named words

  Iterator<String> iter = words.iterator();

  // Now use iterator to print elements in words

  while (iter.hasNext())
  {
    System.out.println(iter.next());
  }
  ```
Some classes implement *Iterator*

- **Look at the Scanner API**
  - It includes the required methods:
    - `hasNext()`
    - `next()`
  - Also provides many in the same vein, for example:
    - `hasNextInt()`
    - `nextInt()`
    - `hasNextLine()`
    - `nextLine()`
    - `etc.`
For each in Java 5

- In many cases, can replace iterator with for-each loop

```java
// Assume an ArrayList<String> named words
// No need to get iterator
// directly use loop

for (String s: words) {
    System.out.println(s);
}
```
For each in Java 5

- **BONUS: it also works for Arrays**

```java
// Assume an array String[] named words
// directly use loop

for (String s: words) {
    System.out.println(s);
}
```
When can I use for-each?

- Must be an *array* or a class that implement the **Iterable** interface
  - Loop up API for **Iterable**
  - What method must an **Iterable** class have?