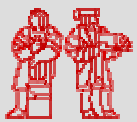
Lecture 9: Lists

MIT-AITI Kenya 2005

In this lecture we will learn….

- `ArrayList` – These are re-sizeable arrays
- `LinkedList brief overview`
- Differences between Arrays and ArrayLists
- Casting
- `Iterator` method (briefly)



Review of Arrays

- Arrays are a simple Data Structure
- Arrays store a row of values of the same type:
  - Primitive types (int, double, etc)

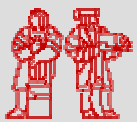
    int[] prices = new prices[10];

    //This stores 10 different prices//

  - Object types (Students, Cars, etc)

    Students[] aitiClass = new Students[70]

    //Each student in the class is a separate object//



Arrays Review Part 2

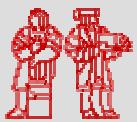
- Access each value in an Array using the index;

  int[] primeNums = new int[20];

  //first 20 prime numbers//

  primeNums[0] = 1; primeNums[3] = 5;
- Remember, Array indices start at 0.

- What was the main problem with Arrays?
  - Array lengths could not be changed once declared.

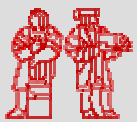


Something better than Arrays?

- As we noticed in the GradeBook Lab, Arrays can be annoying to use due to their fixed length.

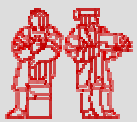
- We need something with a similar structure to that of Arrays, but which can be resized automatically (no more fixed length issues).

- We use the `ArrayList` Class!!!



ArrayList I

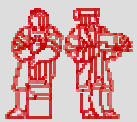
- In an ArrayList, the elements are stored internally as an Array.
- Elements in ArrayLists are stored as type Object
- Thus in order to take an element out of an ArrayList , we will need to cast it into the desired type (we will revisit Casting in future slides)
- Since `ArrayList` is a class, it has its own methods too…



ArrayList II

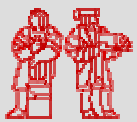
- `get` method is *fast* – it just retrieves the element at the specified index
- `add` method is *slow* – may have to create a larger array and copy over all the elements.

- Other methods in `ArrayList` class include:
  - `set` method – change an entry
  - `size()` method – count number of elements
- To create an ArrayList:
  ArrayList newList = new ArrayList();



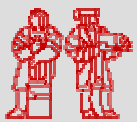
ArrayList III

- ArrayLists are not in the core java language, they are stored in a package
- They must therefore be imported by typing
  - import java.util.*;
    - At the top of your class
- ArrayList
  - ArrayLists have many functional methods that make it easy to use and flexible



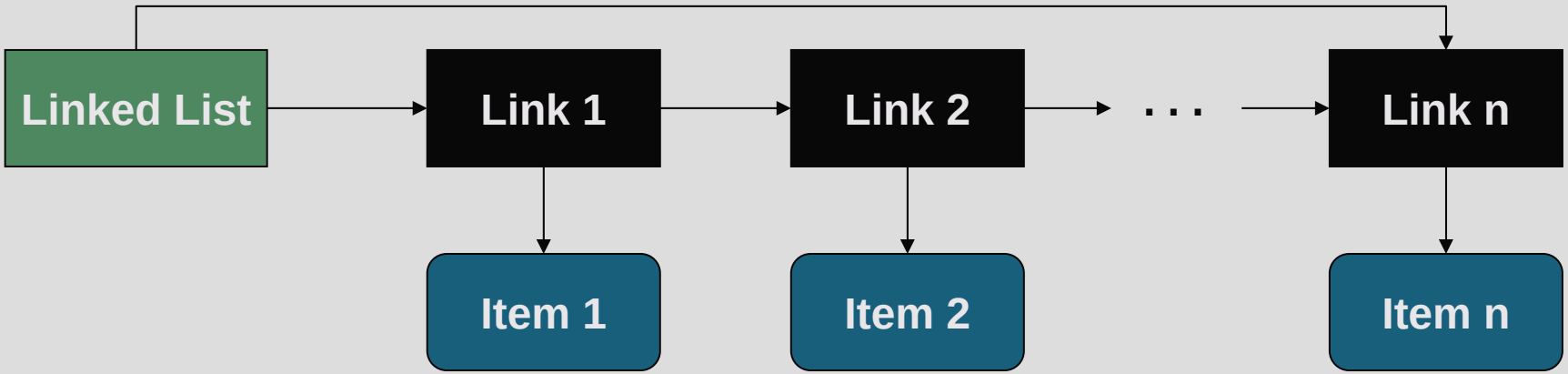
ArrayLists versus Arrays

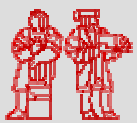
Arrays	ArrayLists
Stores any one type only	Stores only objects (Can store different objects)
Fixed length	Flexible length
Has indices	Does not have indices
Faster at retrieving its contents	Slower at retrieving its contents



Linked List Data Structure

- In a linked list, each link stores an item and is connected to the next link
- The list holds a reference to the first (and maybe the last) element





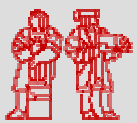
LinkedList

- `A LinkedList` stores its elements internally in a linked list data structure (diagram on previous slide)

  e.g. LinkedList anotherList = new LinkedList();

- `add` method is *fast* – it just appends a new link to the end of the list

- `get` method is *slow* – has to walk down the list retrieve the element at the specified index

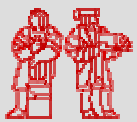


`iterator` method

- Both the `ArrayList` and `LinkedList` classes have an `iterator` method
- `iterator` method returns an object of type `Iterator`
- We use the `iterator` to go sequentially through each of the elements in the list.

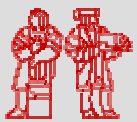
  e.g. for an ArrayList of cars;

  Iterator c = cars.iterator();
- Now c contains all the elements of the *cars* ArrayList, each of type Iterator.



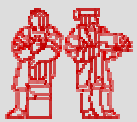
Iterator

- `hasNext` method returns the boolean true when there are more elements to iterate over
- `next` method returns the next element in the iteration
- What is the return type of the `next` method?



Casting I

- When an `ArrayList` uses the `get` method, the return value is of type Object
- We cast this return value from type Object into the actual type it should be
- Casting means *forcing* the type of a value to be changed
- We cast so as to use an object in a different way
- Casting can only be done if the object/primitive being cast is compatible

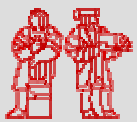


Casting II

- To make a cast, you put the target class name in () and place it before the reference to the object you want to cast

  e.g. double y = 3.14;
  int z = (int) y;

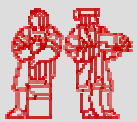
- Another example on next slide…



GradeBook ArrayList example

```
class GradeBook {
   private ArrayList grades;

   public void printGrades() {
  for (int i =0; i<grades.get(i); i++){
      Double g = (Double)grades.get(i);
      System.out.println(g.doubleValue());
      }
   }
}
```



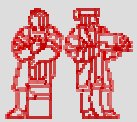
Quiz

- What are the advantages of using an ArrayList?

- What is wrong with the ArrayList implementation below?

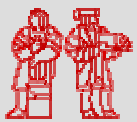
```
ArrayList aList = new ArrayList();
//aList contains several Car objects (remember
  //the car object from yesterday?)
  Car whiteVan = aList.get(matatu);
```

Casting



What have we learned in this lecture?

- How to manipulate `ArrayLists`
- Differences between ArrayLists and Arrays
- Casting (will be useful in future labs)



EC.S01 Internet Technology in Local and Global Communities
Spring 2005-Summer 2005