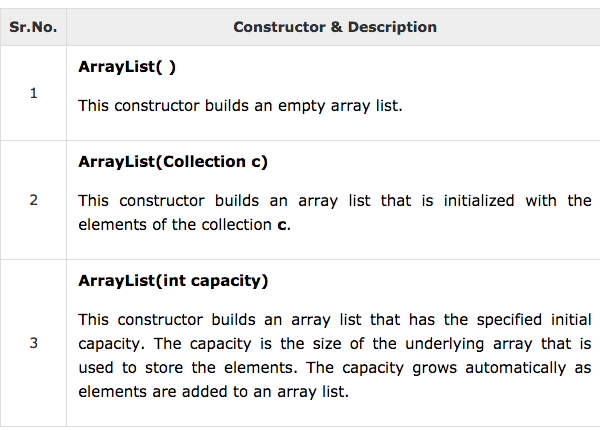
**Week 4**

Review API docs for **ArrayList<E>**, **LinkedList<E>, Iterator<E>**

ArrayList<E> class

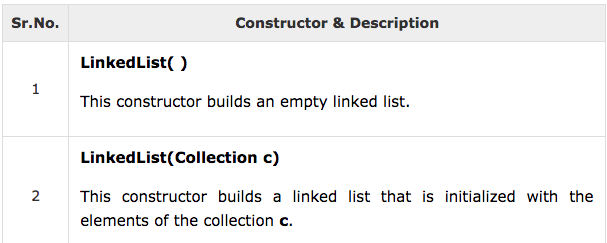
* Extends **AbstractList** and implements the **List interface**.
* ArrayList supports dynamic arrays
  + Standard Java arrays are of a fixed length
* ArrayLists are created with an initial size.
  + When the size is exceeded, the collection is automatically enlarged.
  + When objects are removed, the array may be shrunk
* Constructors



* For methods, see: <https://www.tutorialspoint.com/java/java_arraylist_class.htm>

LinkedList<E> class

* Extends AbstractSequentialList and implements the list interface. It provides a linked-list data structure.
* Constructors



* Methods: <https://www.tutorialspoint.com/java/java_linkedlist_class.htm>

**Use Case**Series of steps where: *A user wants to enrol in a course + they pass a prerequisite of that course*

1. System show list of courses
2. User selects a course
3. User asks System to enrol in course
4. System checks pre-requisites
5. System allocates sessions to user
6. System displays sessions to user

Use cases can have variants.

* E.g. User doesn’t pass pre-requisites, will return an error etc.
* Start off designing the use cases, then dig deeper and create variants

**CRC Cards**

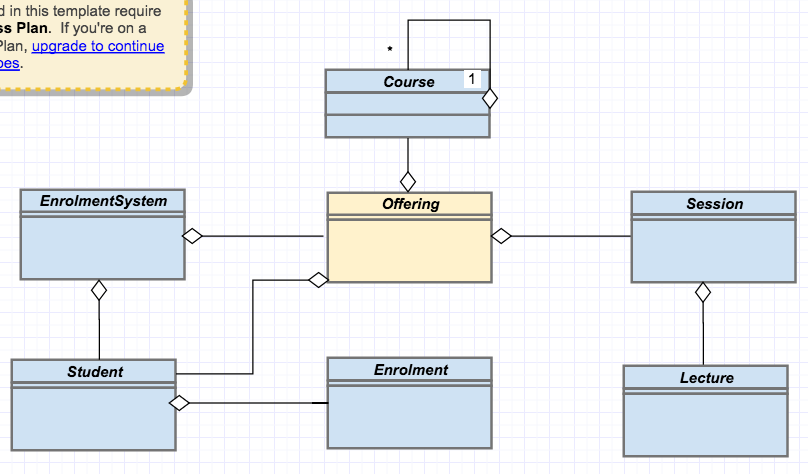
|  |  |
| --- | --- |
| Class | |
| Responsibilities | Collaborators |
|  |  |
|  |  |
|  |  |

**Walkthrough**

1. EnrolSys displays list of courses
2. User selects a course
3. EnrolSys asks Course to enrol student
4. Course gets prereq from Course (“looping” to itself)
5. Course asks Student if pass prereq
6. Courses asks Session to add student
7. Course creates Enrolment object

Not too much inheritance. Only **Lecture inheriting from Session**

UML Diagram for the Walkthrough



Note: Lecture inherits from Session

**Assignment 1**

What to do

* Design before coding
* Follow the process through and make sure you have a design and then code from that design
* Make sure the the code is consistent with the design
* Make sure it passes the tests
* The design can be however you want
* Don’t cheat, don’t look at solutions on git etc.

Make sure you’re reading in from a file.

Set up a file called Input.test and write to system.out for the sake of the marketing system.

Don’t use a package.

Use the default package.

TIP: Make it similar to like the enrolment system?

**UML Sequence Diagram**

Shows interaction between classes.

**Create new Enrolment**

**getPrereq**

**STUDENT**

**STUDENT**

**hasPassed**

**STUDENT**

**COURSE**

­­

**Arrays and Lists**

Arrays are covariant

Lists are not covariant

**Generic Types**

Types with params E, T 🡪 Compiler determines the actual type

Java static type checking 🡪 but Compiler cannot guarantee type safety

Implemented by Type Erasure 🡪 Runtime access only to “raw” types

**Assignment 1**

Design before coding – Follow the OOD process

Submit a UL design doc, not auto generated

UML class diagram should include fields and methods

UML class diagram should conform to code

Don’t look for solutions on github

Only use Stack Over with caution

Make sure ou read from a file args[0], write ot System.out

Make sure main() is in VanRentalSystem.java

Don’t use a package – use only the default package

Make sure the submitted zip file contains files, not in a directory

If your design is a straight line, it is probably wrong.