Java Collection Classes

Com379PT

john.murray@sunderland.ac.uk

Collection Classes

- A collection is a grouping of objects of the same class or sub class
  - Java.util package
  - LinkedList
    - Generic linked list
  - ArrayList
    - Generic dynamic Array

What are Collections?

- Collections represent data items that should be naturally grouped
  - A poker hand (collection of cards)
  - A mail folder (collection of letters)
  - A telephone directory (a collection of name to phone number mappings)

LinkedList & ArrayList

- Two types of Collection class
- Are used in the same way as each other:

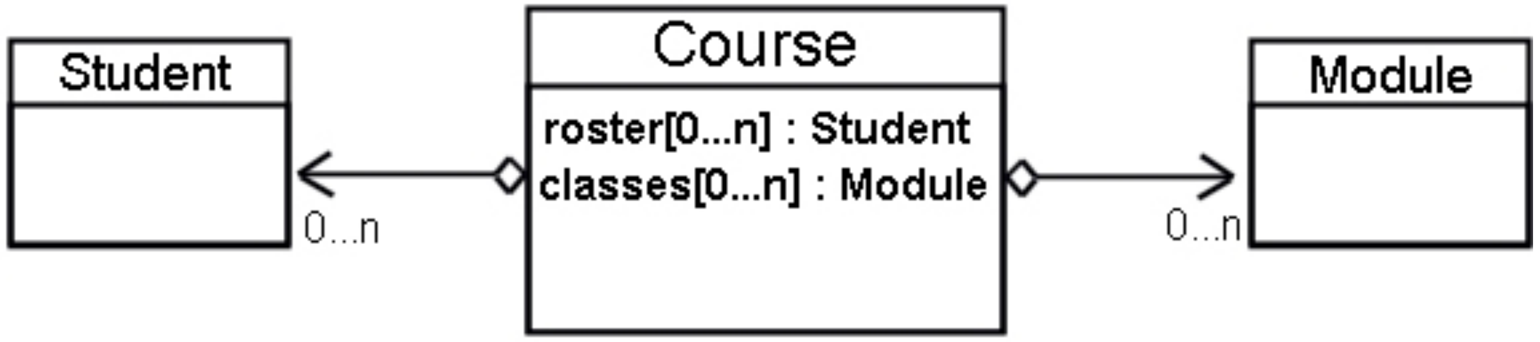
Method	**Description**
void add(Object o)	Add o to the end of the list
void add(int I, Object o)	Add to the i’th position object o
Object get(int i)	Return the i’th object in the list
int size()	Return the size of the list

- Note the type Object is used. All classes extend Object.

Collections Framework

- All collection frameworks contain:
- **Interfaces** – Allow collections to be manipulated independently of their representation.
- **Implementations** – Implementations of the collection interfaces.
- **Algorithms** – Methods for performing searches and sorting.

Collection example



Collection Types

- Array
- List
  - Vector
  - ArrayList
  - Queue
- Associative Array
  - Hashtable
- + More

ArrayList – Collection example

```
public class Student
{
    private String name;

    public Student(String aName)
    {
        this.name = aName;
    }

    public String getName()
    {
        return name;
    }
}
```

ArrayList – Collection example

```
import java.util.ArrayList;

public class Course
{
    private ArrayList studentList = new ArrayList(10);
    private Student aStudent;

    public void addStudent(String aName)
    {
        studentList.add(new Student(aName));
    }

    public Student getStudent()
    {
        aStudent = (Student)studentList.get(0);
        return aStudent;
    }
}
```

ArrayList – Collection example

```
public class TestCourse
{
    public static void main(String [] args)
    {
        Course com379 = new Course();
        com379.addStudent("John");
        Student temp = com379.getStudent();
        System.out.println("Name: " + temp.getName());
    }
}
```

Casting

- Passing one object off as another
- When creating a collection the default type is of Object
- When returning one of the position in the list it is returned of type object

- `aStudent = (Student)studentList.get(0);`

Iterating through the list

- If we have predefined our size we may have some empty elements
- To list all available elements we can iterate

```
Iterator it = list.iterator();
while (it.hasNext())
{
    Student aStudent = (Student)it.next();
    System.out.println(aStudent.toString());
}
```

Why use collections?

- It reduces programming effort?
  - Provides useful data structures
  - Allows you to concentrate in the working of the code
- Allows interoperability among unrelated API's
  - If your communication API has a collection of IP's and my GUI API displays in TABS IP's then they can work together even though they were written separately