CSSE 220 Day 30

Generics
Course Evaluations
Exam Review

Checkout *Generics* project from SVN

Questions

Generic Types

- Another way to make code more re-useful

Before Generics…

- Collections just stored `Objects`
  - Better than creating different collection classes for each kind of object to be stored
  - Could put anything in them because of **polymorphism**
- Used casts to get types right:

  -
    ```
    ArrayList songs = new ArrayList();
    songs.add(new Song("Dawn Chorus","Modern English"));
    …
    Song s = (Song) songs.get(1);
    ```

  -
    ```
    songs.add(new Artist("A Flock of Seagulls"));
    Song t = (Song) songs.get(2);
    ```

Q1

With Generics…

- Can define collections and other classes using type parameters

  -
    ```
    ArrayList<Song> songs = new ArrayList<Song>();
    songs.add(new Song("Dawn Chorus", "Modern English"));
    …
    Song s = songs.get(1); // no cast needed
    ```
  - ~~`songs.add(new Artist("A Flock of Seagulls"));`~~

compile-time error

- Lets us use these classes:
  - in a variety of circumstances
  - with strong type checking
  - without having to write lots of casts

Q2

Example

- Create a doubly linked list
- Include `min()` and `max()` methods
- Use **polymorphism** rather than null checks for the start and end of the list
- Include `fromArray()` factory method

Generics Recap

- Type parameters:
  - `class DLList<E>`
- Bounds:
  - `class DLList<E extends Comparable>`
  - `class DLList<E extends Comparable<E>>`
  - `class DLList<E extends Comparable<? super E>>`
- Generic methods:
  - `public static <T> void shuffle(T[] array)`

Q6, pass in

Course Evaluations

- Your chance to improve instruction, courses, and curricula.

Exam

- Exam is Monday, 6pm here
- Same format as previous exams, a little longer since 2 programming questions
- Comprehensive, but focused on Ch. 13-17, 20

Some Possible Exam Topics

- Simple recursion
- Mutual recursion
- Time-space trade-offs
- Basic sorting algorithms
  - Selection, insertion, merge, and quicksort
  - Efficiency, best/worst case inputs
- Big-oh notation, estimating big-oh behavior of code
- Function objects
- Linked-list implementation
- Basic data structure use and efficiency
  - ArrayList, LinkedList, Stack, Queue, HashSet, TreeSet, HashMap, TreeMap
- Multithreading (not locks)
- Generics