CSSE 220 Day 29

Generics
Exam Review

Checkout *Generics* project from SVN

Questions

Project demo/presentation Thursday

- Informal: no need to dress up
- Think of it as an internal company presentation, not a presentation to the public
- Five-minute presentation, two minutes for questions, two minutes for transition to next team
- Order of teams will be randomly determined
- Turn in your report at the very beginning of class
- Before Thursday, practice getting your computer to work with a New Olin projector

Project demo/presentation Thursday

- Do a *quick* demo of your project
  - Show off any "extra" features or things that work well
- What part was your team's biggest challenge?
- Show one or two interesting code snippets
- Ask for questions
  - **And ask questions of other teams**
- Before Thursday, practice getting your computer working with a New Olin projector

Final Exam

- Exam is Tuesday, Feb. 21 at 6:00 pm
- Same general format as previous exams
- Same resources:
  - Paper part: Three double-sided sheets of paper
  - Computer part: 3 sheets, plus textbook, course web pages and ANGEL pages, JDK documentation, programs in YOUR CSSE220 repository
- Comprehensive, but focused on Chapters 9-18
- May include problems that make sure you understand your team's project code

Final Exam – possible topics

- Interfaces, polymorphism, inheritance and abstract classes
- Exception handling (try, catch, finally, throw, throws)
- OO design and UML class diagrams
- Recursion
- Sorting, searching, Big-oh analysis
- Linked List implementation
- Data structures: List, Stack, Queue, Set, Map (Hash and Tree versions of Set and Map)
- Generic programming
- Event handling, layout managers, exploring the Swing documentation
- Your LodeRunner implementation

Generic Types

- Another way to make code more re-useful

Before Generics…

- Java Collections just stored `Objects`
  - This was better than creating different collection classes for each kind of object to be stored
  - Could put anything in them because of **polymorphism**
- Used class casts to get the 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);
```

run-time error

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

compile-time error

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

Q3-Q5
    ```

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)`

Q6-7, turn in

Some Possible Exam Topics

- Simple recursion
- Mutual recursion
- Time-space trade-offs
- Basic search algorithms
  - Binary search, linear search
  - Efficiency, best/worst case inputs
- Big-oh notation, estimating big-oh behavior of code
- File I/O, exception handling
- Function objects
- Linked-list implementation
- Basic data structure use and efficiency
  - ArrayList, LinkedList, Stack, Queue, HashSet, TreeSet, HashMap, TreeMap
- Multithreading (not locks)
- Generics


LodeRunner Work Time