

# Useful Classes

---

COP3330 - Fall 2006  
University of Central Florida

H. Schwartz

---

## Outline

---

- Generic Classes
- TreeSet
- PriorityQueue
- Enumerations

# Generic Classes

- Classes with type parameters.
  - Generic in the sense that we do not know what type we will be working with.
- Also called "generic polymorphism"
  - Allows a method to do different things (does not involve inheritance in this case)

COP 3330 - Fall 2006

# Generic Classes

- Example (from book):

```
class Wrapper<T> {
    //instance variable
    private T tValue;

    public Wrapper(T t){
        tValue = t; //constructor, t and tValue type are same
    }

    public T get(){
        return tValue; //accessor method
    }

    public void set(T t){
        tValue = t; //mutator method (changes values)
    }

    public String toString(){
        return tValue.toString(); //text representation
    }
}
```

Source: Cohoon and Davidson. Java 5.0 Program Design

COP 3330 - Fall 2006

# TreeSet

---

- Stores a set of objects
  - Keeps contents sorted in ascending order
- Takes in a type parameter
- Performs in good time (add, remove, compare), because internal storage is a tree.

---

COP 3330 - Fall 2006

# TreeSet

---

- Usage Example:

```
...
String stuName1 = "foo bar";
String stuName2 = "OOP Master";
...
TreeSet<String> classRoster = new TreeSet<String>();
classRoster.add(stuName1);
classRoster.add(stuName2);
...
//OOP master decides he/she is not the master
classRoster.remove(stuName2);
...
//check if foo bar really is in the course:
if (classRoster.contains(stuName1)){
...
}
```

---

COP 3330 - Fall 2006

# PriorityQueue

---

- Queue: an ordered structure with one item which may be accessed/removed directly.
  - FIFO
  - LIFO (stack)
  - Based on priority (comparison)
- PriorityQueue: order based on priority  
(Objects in queue must implement Comparable)

---

COP 3330 - Fall 2006

# PriorityQueue

---

- Methods: `<E>` is type
  - `boolean offer(E o)`
  - `boolean add(E o)`
  - `E peek()`
  - `E poll()`
  - `boolean remove (E o)`
  - `int size()`

---

COP 3330 - Fall 2006

# PriorityQueue

- Example:

```
//assume we have a class ToDoItem which implements comparable:
//has a string and priority number, higher = more priority
...
PriorityQueue<ToDoItem> tdList = new PriorityQueue<ToDoItem>();

//populate with things to do:
tdList.offer(new ToDoItem("Cut my hair", 10);
tdList.offer(new ToDoItem("Teach a course", 2);
...
//find out next thing to do
ToDoItem nextTD = tdList.poll();
while (nextTD != null){
    System.out.println(tdList); //calls toString to print item
    waitForKey();
    nextTD = tdList.poll();
}
```

COP 3330 - Fall 2006

# Enumerations

- class Enum<T>
  - super class of all "enums"
  - enums are built in as part of syntax
- Good for storing a list of things which could be a class, but wouldn't have methods.
  - Comparing with self

COP 3330 - Fall 2006

# Enumerations

---

- Example:

```
//list of topics in OOP
//just inside a class
public enum Topic {HERITANCE, ENCAPSULATION, OBJECTS, CLASSES, ABSTRACTION};
...

//when using (in method):
Topic todayTopic = Topic.ENCAPSULATION;
...
if (todayTopic == Topic.OBJECTS){
    ...
}
```