CS 335 — Software Development

The Chain of Responsibility Pattern

April 21, 2014

Exception handling

```
try {
  Scanner file = new Scanner(new FileInputStream("operations.dat"));
  while (file.hasNext())
     Class.forName(file.nextLine());
} catch (IOException ioe) {
  System.out.println("Couldn’t find file operations.dat");
  System.exit(-1);
} catch (ClassNotFoundException cnfe) {
  System.out.println("Couldn’t load operation " + cnfe.getMessage());
  System.exit(-1);
}
```

Pattern matching

```
datatype treeGenus = Oak | Elm | Maple | Spruce | Fir |
                     Pine | Willow;

datatype treeDivision = Deciduous | Broadleaf;

fun division(Pine) = Coniferous
  | division(Spruce) = Coniferous
  | division(Fir) = Coniferous
  | division(x) = Broadleaf;
```

Recursive linked-list methods

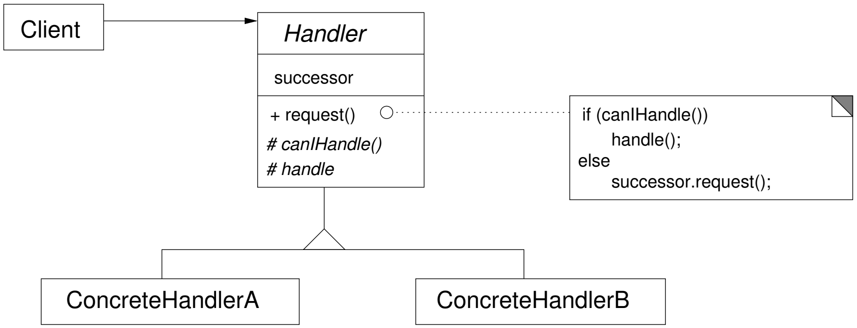
```
public class Node {

   private int datum;
   private Node next;

   public boolean contains(int item) {
      if (item == datum) return true;
      else if (next != null) return next.contains(item);
      else return false;
   }
}
```

Chain of Responsibility intent

> *Avoid coupling the sender of a request to its reciever by giving more than one object the chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.*

Gamma et al, pg 223

Chain of Responsibility structure



```
 if (canIHandle())
        handle();
else
        successor.request();
```

Based on Gamma et al, pg 224.