Patterns, Set, and Maps

4-15-2011
Opening Discussion

- Minute essay comments:
  - Material on final exam.
  - Simple description of a pattern. Something you probably did in math in elementary school was looking at repeating patterns.
  - A pattern specifies a certain form for something. A match occurs if a value fits the given pattern.
We have used three types of patterns previously:

- Value literals
- Tuples
- Type matches

The last two start to show the power of pattern matching. In particular, they show that part of a pattern can be a variable name that binds to part of the pattern.
Variable Binding

- When a pattern is matched, any words that start lower case are assumed to be variable names you want bound.
- Use an underscore for anything you want to match stuff, but ignore the value.
- Use @ to bind a name to a match you are also further specifying.
- To match the value of an outside variable put the variable name in backticks.
You can use patterns to pull out parts of XML or match on different types of nodes.

Simply put the variable names you want inside of curly braces.

- val <a>{s}</a> = node
Case Class Patterns

- The real power of case classes in Scala comes from the fact they can be used in matches.
  - stu match {
    - case Student(n,q,t,a) => ... 
  }
- You can do this type of matching on events to pull out the fields you care about if you don't want the full event.
  - case MouseMoved(source,point,mod) => ...
You can also make patterns with collections.

- case Array(a,b,c) => // use a, b, and c

Even more cool is what you can do with Lists.

- case h::t => // h is head and t is tail
- case a::b::Nil => // two element List

This can be ideal for recursive methods on lists.

```scala
def len(lst:List[Int]) = lst match {
  case Nil => 0
  case h::t => 1+len(t)
}
```
Patterns are used in a lot of places in Scala, not just cases and matches.

The initial declaration of variables is a pattern match. That is why we could assign from tuples.

The “variable name” in a for loop is actually a pattern. If the pattern isn't matched by an element, that element is skipped.
Sets, Maps, and Buffers

- The Scala collections library is a lot richer than just Lists and Arrays.
- I want to introduce three other types of collections to you as they can make your life a lot easier for certain tasks.
- They are all parametric so they can work on a variety of types.
Sets

- This is a collection that isn't ordered and doesn't allow duplicates.
- There are both mutable and immutable sets. By default you get the mutable version.
Buffers

- A buffer is a sequence, like an array or a list, but it is mutable like an array and grows like a list.
- You find these in the scala.collection.mutable package.
Maps

- This collection type has two type parameters for a key and a value type.
- You store values and look them up by key.
- The keys are unique.
- There are both mutable and immutable maps. By default you get the mutable version.
Minute Essay

- What questions do you have about stuff?