CS61B Lecture #35

Today:  Enumeration types

Coming Up:  Graph Structures: DSIJ, Chapter 12
Side Trip into Java: Enumeration Types

• Problem: Need a type to represent something that has a few, named, discrete values.

• In the purest form, the only necessary operations are == and !=; the only property of a value of the type is that it differs from all others.

• In older versions of Java, used named integer constants:

```java
interface Pieces {
    int BLACK_PIECE = 0, // Fields in interfaces are static final.
    BLACK_KING = 1,
    WHITE_PIECE = 2,
    WHITE_KING = 3,
    EMPTY = 4;
}
```

• C and C++ provide enumeration types as a shorthand, with syntax like this:

```java
enum Piece { BLACK_PIECE, BLACK_KING, WHITE_PIECE, WHITE_KING, EMPTY };
```

• But since all these values are basically ints, accidents can happen.
Enum Types in Java

• New version of Java allows syntax like that of C or C++, but with more guarantees:

```java
public enum Piece {
    BLACK_PIECE, BLACK_KING, WHITE_PIECE, WHITE_KING, EMPTY
}
```

• Defines Piece as a new reference type, a special kind of class type.

• The names BLACK_PIECE, etc., are static, final enumeration constants (or enumerals) of type PIECE.

• They are automatically initialized, and are the only values of the enumeration type that exist (illegal to use new to create an enum value.)

• Can safely use ==, and also switch statements:

```java
boolean isKing (Piece p) {
    switch (p) {
        case BLACK_KING: case WHITE_KING: return true;
        default: return false;
    }
}
```
Making Enumerals Available Elsewhere

- Enumerals like `BLACK_PIECE` are static members of a class, not classes.
- Therefore, unlike C or C++, their declarations are not automatically visible outside the enumeration class definition.
- So, in other classes, must write `Piece.BLACK_PIECE`, which can get annoying.

However, with version 1.5, Java has static imports: to import all static definitions of class `chess.Piece` (including enumerals), you write

```java
import static chess.Piece.*;
```

among the import clauses.

- Alas, cannot use this for enum classes in the anonymous package.
Operations on Enum Types

- Order of declaration of enumeration constants significant: `.ordinal()`
gives the position (numbering from 0) of an enumeration value. Thus,
Piece.BLACK_KING.ordinal () is 1.

- The array `Piece.values()` gives all the possible values of the type.
Thus, you can write:

```java
for (Piece p : Piece.values ())
    System.out.printf ("Piece value #d is %s%n", p.ordinal (), p);
```

- The static function `Piece.valueOf` converts a String into a value of
type `Piece`. So `Piece.valueOf ("EMPTY") == EMPTY`. 

Fancy Enum Types

- Enums are classes. You can define all the extra fields, methods, and constructors you want.

- Constructors are used only in creating enumeration constants. The constructor arguments follow the constant name:

```java
enum Piece {
    BLACK_PIECE (BLACK, false, "b"),
    BLACK_KING (BLACK, true, "B"),
    WHITE_PIECE (WHITE, false, "w"),
    WHITE_KING (WHITE, true, "W"),
    EMPTY (null, false, " ");

    private final Side color;
    private final boolean isKing;
    private final String textName;

    Piece (Side color, boolean isKing, String textName) {
        this.color = color; this.isKing = isKing; this.textName = textName;
    }

    Side color () { return color; }
    boolean isKing () { return isKing; }
    String textName () { return textName; }
}
```