Enumerations

Basic Enumerations

```
public enum Switch { ON, OFF };
```

- Creates a new data type called "Switch"
  - So I can declare variables of this type, e.g. `Switch hallLight;`
- Creates new literals for each enumerated value
  - So I can assign values, e.g. `hallLight=Switch.ON;`
  - So I can compare values, e.g. `if (hallLight==Switch.OFF) { …`
- Variables of this type can *only* have enumerated values

Basic Enumerations – Under the Covers

Can use == because there is only 1 ON and 1 OFF object

```
public enum Switch { ON, OFF };
```

- Creates a new ~~data type~~ class called "Switch"
  - So I can declare reference variables of this type, e.g. `Switch hallLight;`
- Creates new ~~literals~~ static variables for each enumerated value
  - So I can assign values, e.g. `hallLight=Switch.ON;`
  - So I can compare values, e.g. `if (hallLight==Switch.OFF) { …`
  - All valid objects in this class are pre-declared as static final variables
  - Therefore, no constructor is required… just reference existing objects
- Variables of this type can *only* have enumerated values

Enumerations extend Enum

- Enum is a Java library class
  - w/ methods: name(), ordinal(), compareTo(), values()
- hallLight.name() returns String "OFF" or "ON"
- hallLight.ordinal() returns int 0 or 1
  - (or 2 or 3… for larger enums)
- hallLight.compareTo(Switch other) compares ordinals
  - (returns int <, 0, >)
- values() returns [Switch.OFF, Switch.ON]
  - An array of Switch objects in ordinal order

Enhancing Enumerations

- Since an enum is really a class under the covers, we can add other fields and methods to an enum
- The list of values becomes a list of constructor invocations
- Enables enumerations to be much more sophisticated than just a simple set of values

- It is also possible to extend an enum
  - e.g. ElectricSwitch vs. Valve (water switch)

Enumeration Syntax

Cannot "extend" because implicitly extends Enum

```
modifiers enum name attributes {
    contents_list;
    fields_and_methods
}
```

Optional

- *contents_list* : comma separated list of enumeration values
  - By convention, enumerated values are all uppercase (like constants)
  - May be literals, or may contain parenthesis and constructor arguments
    - May even be followed by { anonymous sub-class }