Interfaces

Abstract classes

\[ \rightarrow \text{cannot be instantiated} \]
\[ \rightarrow \text{can have concrete methods} \]
\[ \rightarrow \text{usually have abstract methods} \]
Java uses single inheritance and this sometimes can cause difficulties.

Creating "kitchen-sink" classes

Some languages use multiple inheritance, but that also has issues.
A Java class can implement multiple interfaces.

An interface just specifying what methods a class must have. It does not implement code.
Interfaces are rules or behaviors aka mixins

```java
public interface Enumeration {
    boolean hasNext();
    Object nextElement();
}
```
What is an enumeration?
It's a listing of the elements in a collection.

```
Enumeration e = somehow get enumeration
while (e.hasNext()) {
  Object o = e.nextElement();
  // do something w/o
}
```
java.util contains

the Java Collections Framework

(JCF)

Enumeration
Two Way Enumeration → set direction

Min Max → set smallest/largest object
Smallest/Largest (finite) set of #s

0 < 1 < 2 < ... < 10

We know how to compare #s
What about sets?

$A \leq B \leq C \ldots \leq D$

What about letters?

$q < b < c < \ldots < z < A < B < \ldots$

This is important for sorting
Shallow Copies vs Deep Copies

GroceryList g = new GroceryList();
g.add(new Apple());
g.add(new Poptarts());
g.add(new BrusselsSprouts());
GL

GroceryList e = gj

GL e

ShallowCopy

e.add(new DietRokp());

Apple

Pepita

BrusselSpr

j.remove(DC)
```
GroceryList\ e = g.clone();
```

```
GL\ e
```

```
GL e
```

```
GL \&
```

```
\&GL
```

```
App\ &
```

```
P1\ &
```

```
BS\ &
```

```
A&g
```

```
P7\ &
```

```
G5\ &
```

```
```

```
```

```
```

```
```

```
```
A little handwaving ahead ....

```java
public class GroceryList {
    public Enumeration getItems() {
        return ...;
    }

    public GroceryList clone() {
        GroceryList nl = new GroceryList();
        Enumeration e = getItems();
        while (e.hasMoreElements()) {
            //
        }
    }
```
Object e = e.nextElement();

l.add (e.clone());

return l;
GL g = new GL();

GL g

GL

g.add(new Apple());

Apple
\[ G \cdot L = \text{g.clone();} \]

// 1. Create a new GL

\[ A, B \rightarrow C \]

// G x = \text{gGLfloat}();
1. Cloneable interface

```java
public interface Cloneable {
}
```

Marker interface
public class Object {

    public Object clone() {
        return null;
    }
}
Fly weights

(Design Pattern)

\[
\frac{GL}{\text{clone}}
\]

\[
\frac{G-I}{\text{clone}}
\]

\[G-I = \text{Clone system}\]
\[ b = g . \text{clone}(i) \]