Abstract Classes
The Real Story?

```
NaturalNumber
  implements NaturalNumber1L
  implements NaturalNumber2

NaturalNumber-
  Kernel
  extends Standard
  extends Comparable

Standard

Comparable
```
The Real Story?

Does each of NaturalNumber1L and NaturalNumber2 contain code for all 21 (yes, there are 21) methods that are introduced in, or inherited by, interface NaturalNumber?

No!
The Ubiquitous Class: `Object`

• *Every* class in Java extends `Object`, which is a special built-in class that provides default implementations for the following instance methods (among a few others that are not so important):

  ```java
  boolean equals(Object obj)
  int hashCode()
  String toString()
  ```
equals

boolean equals(Object obj)

• Reports whether this is equal to obj.
• Ensures:

  equals = (this = obj)
boolean equals(Object obj)
• Reports whether this is equal to obj.
• Ensures:
  \[ \text{equals} = (\text{this} = \text{obj}) \]

The default implementation in object checks reference equality, though we expect object value equality! So, we (almost always) need to override this method.
hashCode

```java
int hashCode()
```

- Returns an `int` value that is functionally determined by the value of `this`.
- Ensures:
  
  \[
  \text{hashCode} = [\text{result of some fixed integer-valued function applied to } \text{this}]
  \]
hashCode

int hashCode()

• Returns an int value that is functionally determined by the value of this.
• Ensures:

hashCode = [result of some fixed integer-valued function applied to this]

The default implementation in Object returns an int that depends on the reference value of this, though we expect it to depend on the object value! So, we (almost always) need to override this method.
**toString**

`String toString()`

- Returns the string representation of `this`.
- Ensures:
  
  `toString = [the string representation of this]`
toString

- Returns the string representation of this.
- Ensures: 

  `toString = [the string representation of this]`

The default implementation in `Object` returns a `String` that shows the reference value of this, though we expect it to show the object value! So, we (almost always) need to override this method.
Abstract Classes

• Java permits you to write a kind of “partial” or “incomplete” class that contains bodies for some but (typically) not all of the methods of the interfaces it claims to implement.

• Such a class is called an *abstract class*:

```java
abstract class AC implements I {
    ...
}
```
• Java permits you to write a kind of “partial” or “incomplete” class that contains bodies for some but (typically) not all of the methods of the interfaces it claims to implement.

• Such a class is called an abstract class:

  ```java
  abstract class AC implements I {
    ...
  }
  ```

Because some methods still might not have bodies, Java will not let you instantiate an abstract class; that is, you cannot use an abstract class like a normal class and create a new object from it.
public abstract class NaturalNumberSecondary implements NaturalNumber {
...
}

\textbf{Example}
public abstract class NaturalNumberSecondary implements NaturalNumber {
...
}

This relationship is implicit: every class that does not extend some other class directly extends Object.
NaturalNumberSecondary

- This abstract class has code that overrides the default implementations (inherited from Object) of equals, hashCode, and toString for NaturalNumbers, so they do “the right thing”, i.e., so their behaviors are based on object values rather than reference values
  - Details of this code later... see “Resources”
NaturalNumberSecondary

- This abstract class also has code that implements all the methods *introduced* in the NaturalNumber interface, but not those inherited by it from other interfaces
  - Details of this code later... see “Resources”
  - Note that this still leaves the methods introduced in the Standard and NaturalNumberKernel interfaces without bodies; hence, it’s an abstract class
The Bigger Picture: Old Story

```
   NaturalNumber1L
     |       |
     v       v
    NaturalNumber
      |       |
      v      v
     Comparable
      |      |
     Standard

  extends
      ^
      \   
      |   |
      v  v
    NaturalNumber-Kernel

  extends
      ^
      \   
      |   |
      v  v
    NaturalNumber

  implements
      ^
      \   
      |   |
      v  v
  NaturalNumber1L

  implements
      ^
      \   
      |   |
      v  v
  NaturalNumber2
```
The new abstract class plus Object, in context...

The Bigger Picture: New Story

The new abstract class plus **Object**, in context...

```
NaturalNumber1L
    extends
    implements
    NaturalNumberSecondary
    extends
    NaturalNumber
        extends
        implements
        Standard
            extends
            Comparable
                extends
                Object
```

29 August 2016
OSU CSE
The Bigger Picture: New Story

NaturalNumber

- newInstance
- clear
- transferFrom

NaturalNumberKernel

- extends
- implements

NaturalNumber

- constructors
- multiplyBy10
- divideBy10
- isZero
- increment
- ... root
- extends
- implements

Standard

Comparable

Object

NaturalNumberSecondary

extends

NaturalNumber1L

extends

NaturalNumber2

extends
The Kernel Classes

• This leaves the kernel classes (in the example, `NaturalNumber1L` and `NaturalNumber2`) with only a few things left to implement, i.e., the 4 constructors plus the 6 methods introduced in:
  − Standard (i.e., `newInstance`, `clear`, and `transferFrom`)
  − `NaturalNumberKernel` (i.e., `multiplyBy10`, `divideBy10`, and `isZero`)
Has *layered* method bodies for `equals`, `hashCode`, and `toString` (overriding those methods from `Object`) ...
Layered Methods

... and *layered* method bodies for all 15 methods introduced in `NaturalNumber`.

![Diagram showing class inheritance and method bodies](image-url)
Kernel Methods

Has bodies for 4 constructors plus 6 methods introduced in Standard and NaturalNumberKernel.
Kernel Methods

Has bodies for 4 constructors plus 6 methods introduced in Standard and NaturalNumberKernel.
Factoring Out Common Code

• Method bodies that can be written once—and work for any implementation of \texttt{NaturalNumberKernel} because they are \textit{programmed to that interface}—have been \textit{factored out into an abstract class}

• This leaves only constructors and a few kernel methods to be implemented in \texttt{NaturalNumber1L}, \texttt{NaturalNumber2}, and future kernel classes (if any)
Factoring Out Common Code

• Method bodies that can be written once—and work for any implementation of NaturalNumberKernel because they are *programmed to that interface*—have been *factored out into an abstract class*

• This leaves only constructors and a few kernel methods to be implemented in NaturalNumber1L and future kernel classes (if any)

*This is a best practice use for abstract classes.*
Another Kettle of Fish

• The code in each *kernel class* (e.g., in the example `NaturalNumber1L` and `NaturalNumber2`) implements only 4 constructors and 6 methods each, not all 21 methods of `NaturalNumber`

• But, implementing these few methods is different than implementing the other 15 *layered* methods where you can *call* the kernel methods to do the work!
Resources

• OSU CSE Components API: NaturalNumber
  – http://cse.osu.edu/software/common/doc/