Kernel Implementations II
A Second Kernel Class Example

• Suppose you wish to design a kernel class that implements `NaturalNumberKernel`

• What existing components (including built-in types of Java, and the Java libraries) could you build it on top of?
  – In other words, what could you use as a data representation that could be interpreted as a `NaturalNumber` value?
NaturalNumber Family

Has bodies for the 4 constructors, plus all 6 methods introduced in Standard and NaturalNumberKernel.
NaturalNumber Family

- newInstance
- clear
- transferFrom

constructors
- multiplyBy10
- divideBy10
- isZero

- extends
  - Standard
  - NaturalNumber-Kernel
    - extends
      - Comparable
      - Object
    - implements
      - Standard
    - extends
      - NaturalNumberSecondary
    - extends
      - NaturalNumberAsSomething
Design Trade-offs

• As the implementer of a kernel class, it is entirely up to you how to represent every value allowed by the mathematical model in the kernel interface!

• Some designs will make the code for the methods you need to write:
  – Easier to understand and make correct
  – More efficient
  – ...

27 June 2013
OSU CSE
Design Trade-offs

• As the implementer of a kernel class, it is entirely up to you how to represent every value allowed by the mathematical model in the kernel interface!

• Some designs will make the code for the methods you need to write:
  – Easier to understand and make correct
  – More efficient
  – ...

For a first kernel implementation, this should be the most important criterion: get something working.
Design Trade-offs

• As the implementer, it is entirely up to you how you represent every value allowed by the mathematical model in the kernel interface.

• Some designs will make the code for the methods you need to write:
  – Easier to understand and make correct
  – More efficient
  – ...

For later kernel implementations, this might be the most important criterion.
Represent It Using a String?

• Suppose you decide to represent a NaturalNumber as a String whose characters are the digits of that number
• Such a kernel class might be called NaturalNumberAsString
Client View

• A situation as seen by a client of the kernel class `NaturalNumberAsString`, based on its interface:

```
this
165
```
Implementer View

- The same situation as seen by the kernel class implementer, when the data representation is a single `String` called `rep`:
Client View

• A situation as seen by a client of the kernel class `NaturalNumberAsString`, based on its interface:

\[\text{this} \rightarrow 0\]
Implementer View

• The same situation as seen by the kernel class implementer, when the data representation is a single `String` called `rep`:

```
"0"
```
Implementer View

The same situation as seen by the kernel class implementor, when the data representation is a single String called `rep`:

Is this really a good idea? Or should the representation of \( 0 \) be ""? Why or why not?

"0"
Represent It Using an Array?

• Suppose you decide to represent a \texttt{NaturalNumber} as an array of \texttt{ints} that are the digits of that number

• Such a kernel class might be called \texttt{NaturalNumberAsArray}
Client View

• A situation as seen by a client of the kernel class `NaturalNumberAsArray`, based on its interface:
Implementer View

• The same situation as seen by the kernel class implementer, when the data representation is an array of integers called rep:

```
1       6       5
```
Client View

• A situation as seen by a client of the kernel class `NaturalNumberAsArray`, based on its interface:

```text
this
```

```text
0
```
Implementer View

• The same situation as seen by the kernel class implementer, when the data representation is an array of \texttt{ints} called \texttt{rep}:
Implementer View

The same situation as seen by the kernel class implementer, when the data representation is an array of *ints* called *rep*:

```
rep:

0
rep[0]
```

Is this really a good idea? Or should the representation of 0 be something else? Why or why not?
Represent It Using a *string*?

- Suppose you decide to represent a `NaturalNumber` as a *string of integer* whose entries are the digits of that number.
- Note that here we are thinking of a possible *mathematical model* of the representation, not in terms of a specific Java component family.
Represent It Using a *string*?

- Suppose you decide to represent a `NaturalNumber` as a *string of integer* whose entries are the digits of that number.
- Note that here we are thinking of a possible mathematical representation, not in terms of a specific Java component family.

An advantage: There are several component families with this mathematical model, and we could use any of them that has the “right” methods for what we need to do.
Client View

• A situation as seen by a client of the yet-to-be-named kernel class, based on its interface:
Implementer View

• The same situation as seen by the kernel class implementer, when the data representation is a string of integer called rep:

\[
\text{this} \rightarrow \text{rep} \rightarrow \langle 1, 6, 5 \rangle
\]
Which component family with a string of integer model would be best to use: Queue, Sequence, or Stack?
(Hint: Think about methods to be implemented, and how well they match the methods available for the candidate representation.)
Implementer View

- With Stack, how about this different representation for the number 165:
Represent It Using a Stack

- It turns out that the “reversed” string representation works out very well when the type of rep is Stack<Integer>
- The code for this is in NaturalNumber2 – See “Resources”
Let’s Look at
NaturalNumber2.java
Resources

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