```java
if (#CLAs is 1) {
    try {
        n = Integer.parseInt(args[0]);
    } catch (NumberFormatException e) {
        Print usage MSG
        quit.
    } else if (#CLAs is 2) {
```
check if args[0].equals("-v") is true.
if its false: usage()
t-

n = Integer.parseInt(args[1])

} catch (NumberFormatException e) {
    System.out.println("usage");
}
Next permutation:

Say \( n = 8 \)

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\times & 6 & 5 & 4 & 3 & 2 & 1 & & \\
\hline
\end{array}
\]

\[\xrightarrow{\text{reverse}}\]

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\times & 6 & 5 & 7 & 1 & 2 & 3 & 4 & 8 \\
\hline
\end{array}
\]

---

\[L3\]
Default action of `toString()`: print the address in JVM at the object its called on.

```
    a 1  -->  "Dick"
       |                   |
       |                   |
```

header of `toString()` in object `x`:

```java
public String toString()
```
Access levels in Java:

- Private
- Package (default)
- Protected
- Public

Rule:
- When overriding an inherited method, you may not narrow its access level.
Over-riding vs. Over-loading an inherited method.

To **over-ride**: heading
use same func in ancestor class, do not narrow access.

To **over-load**: use a different func heading as in Ancestor.
header  at  equals in Object:

public boolean equals(Object x) 

default action at equals:

\[ a \rightarrow \text{"Dick"} \quad e \rightarrow \text{"Dick"} \]

returns true iff addresses in JVM are same.
- **Note:**

- Compare primitive types for equality use `==`

  \[ x == y \]

- Compare reference types for equality use `equals()`

  \[ a.equals(b) \]
General template for overriding equals(): in class Blah

```java
public boolean equals(Object x)

    Blah B;
    boolean eq = false;
    if (x instanceof Blah)

        B = (Blah)x;

    // compare this to B
    // field by field, and
    // assign eq accordingly

    return eq;
```
Data hiding:

- use private access modifiers to restrict access to internal state of the new data type.

- two classes of methods
  
  - access functions return info about state of an object
  
  - manipulation procedures alter the internal state