Immutable Objects

* They never change (at least as viewed from the outside)
* Excellent for multiple threads: because they never change after construction (no corruption risks, no dirty read risks)
* It is ***necessary*** to declare the class to be **final**.
  + This prevents a subclass from being written and simulating change.
* Fields **may** be declared **final**, but don’t have to be.
* No mutators (setters)... nothing can be changed after construction.
* Fields can be:
  + Primitive data types: **boolean**, **char**, **byte**, **short**, **int**, **long**, **float**, and **double**.
  + Other immutable types: **String**, **BigDecimal**, **Integer**, **Double**, etc.
  + Copies of mutable objects and copies are created and returned. A hassle, but the only way to be safe.
  + Arrays of primitive types or arrays of immutable types
    - the array must be **clone()**'d upon construction.
    - if the array is to be returned, create a **clone()**.
    - If array length is zero, then it’s immutable! (and no clone is needed).