CH 13

Invariant

↳ property of a program that is always true

↳ to make an invariant hold:

↳ make the invariant true for the initial state

↳ Creators / producers

↳ ensure that all changes to the object keep the invariant true

↳ observers / mutators

Abstraction Function

$AF : R \rightarrow A$

R: rep values

A: abstract values

Rep Invariant

$RI : R \rightarrow boolean$

RI(r) TRUE for $r \in R$ IFF $AF : R \rightarrow A$

- **Every abstract value is mapped to by some rep value** . The purpose of implementing the abstract type is to support operations on abstract values. Presumably, then, we will need to be able to create and manipulate all possible abstract values, and they must therefore be representable.
- **Some abstract values are mapped to by more than one rep value** . This happens because the representation isn't a tight encoding. There's more than one way to represent an unordered set of characters as a string.
- **Not all rep values are mapped** . In this case, the string "abbc" is not mapped. In this case, we have decided that the string should not contain duplicates. This will allow us to terminate the remove method when we hit the first instance of a particular character, since we know there can be at most one.