

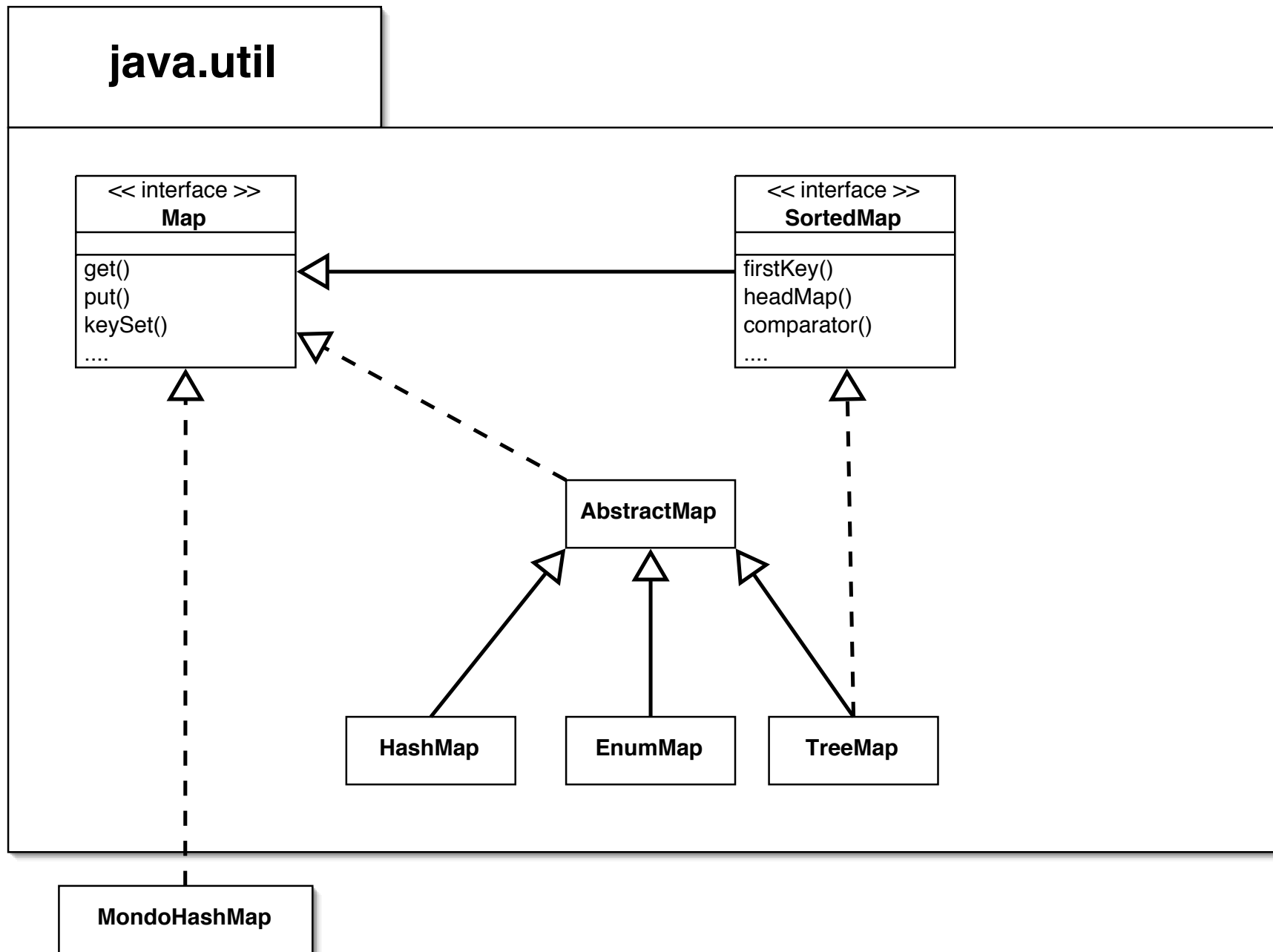
# Interfaces and all that; Thoughts on Hashing

Ch. 6

# Quiz directions

- Individual section
  - Circle correct answer for each question
  - You get 2 tries for each question
    - Correct on 1st try (top line): 3 pts
    - Correct on 2nd try (bottom line): 1 pt
  - Keep your answers! (On quiz sheet)
  - Hand in sheet when complete
- Group section
  - Start when all members of group are done w/ individual
  - Again, 2 tries
  - Grade yourselves as you go along
  - Hand in final group sheet

# The Map hierarchy



# Why all the complexity?

- Why two interfaces, an abstract class, and a bunch of concrete classes?
- Common OO design principle
- **Interface**: provides a *contract* or *specification* of what an object should be able to *do*
  - Has no data of its own
- **AbstractClass**: provides “most” of the implementation of an interface
  - Takes grunt work out of instantiating an interface
  - May not be the most efficient possible
- **Concrete Class**: provides everything necessary to use

# Interchangability

- Common view of interfaces:
  - “Interfaces are Java’s kludge to get around the lack of multiple inheritance”
  - **WRONG!**
  - (mostly)
- Interfaces describe *properties* of a thing -- how you can *use* it
- Any two objects that obey the same interface are (should be) *semantically interchangeable*
  - Can plug either in to same place and have function of program remain same
- They may support diff extra stuff, side effects, have different efficiencies, etc.

# Using interfaces

- Can't call "new" on an interface
- Interface isn't a full-fledged class
- Can declare a variable to be of an interface type
  - `Map localMap=new MondoHashMap( ) ;`
- Note: thing on left can be an interface, thing on right *must* be a concrete class (or something that produces an object of a concrete class)
- Can always refer to an object by one of its interfaces