

Generics Reprised, Serialization, More Code Review

L05, Sep 8, 2008

Generic gotchas

- Generics are purely *compile time* syntactic convenience
- Does not change data structure *at all*
 - Makes *no difference* at runtime
- All data stored internally as type “Object”
 - Typecasts inserted by the compiler

Serialization

- How do you save and restore objects?
- `java.io.Serializable`
- implement the `Serializable` interface (really nothing to do)
- Use `ObjectInputStream` and `ObjectOutputStream`

Serialization caveats

- What happens when a superclass isn't serializable?
- What happens when your state is awkward to serialize?
- use `readObject` and `writeObject`
- if super class is `Object` don't worry about it
- Some things can't be serialized (e.g. `tread`) mark it "transient"

Serialization gotcha

- `ObjectOutputStream` keeps a reference to objects written
- It **WILL NOT** resave an object
- Be sure you don't make changes after a serialization and expect them to be saved
- You could use `ObjectOutputStream.reset()` but...

More

- You can name fields (keep track of what's what)
- Versioning: what if you change the class?
- Externalization versus Serialization (DIY)
- Use serialized objects over network
- Variable name affect size of serialized object!

Code review!
