hashCode, Nested Classes and enum

Monday, February 19, 2018 10:05 AM

hashCode

- Items are deterministically hashed (need not search through entire collection)
- Referencing hashcode vs. array index?

`hashCode()`

- Returns an int
- Two keys/ objects which are the same (.equals()) must have the same hashCode()
  - Have to overwrite Object hashCode() in set ADT
  - Sets check for hashCode
  - Put fields into an array
    - `return Arrays.hashCode(a);`
    - For different types: can put in multiple arrays and XOR hashcodes together (fairly equal distribution of 0s and 1s)
- Two objects with the same hash code do not have to be equal

Nested Classes

- Groups logically relevant classes together
- Nested helper class has no use outside container class (usually declared as private)
- Nested class = can access private fields and methods
- Static/ non-static (inner class)

Local Class

- Declare a class within a function
- Definition and usage of class closer together
- e.g. Comparators

Variable capture

- Local class makes a copy of local variables from enclosing method

Effectively final

- Local classes cannot modify captured variables
- Java only allows access to variables explicitly declared as final OR effectively final (not modified once initialised)

Anonymous Class

- Declared and instantiated in a single statement (one-time use)
- `new X (arguments) { body }`
  - X = class/ interface (cannot be both)

enum

- A type, like classes and interface
- Type-safe
  - Each constant of enum type is an instance of the enum class
  - `public static final`

Custom Methods

- Constant-specific class body

    - Else, cannot inherit from enum (final)
- Each constant = anonymous class

Inheritance

- All enum inherits from Enum implicitly
- enum hence cannot extend from other classes
- Recursive generics from Enum<Type>
    - Enum<E extends Enum<E>>
- Contains enum fields

`static { … }`

- Static initialiser, called when class is first used

`EnumSet, EnumMap`