CMSC 132: OBJECT-ORIENTED PROGRAMMING II

Miscellaneous

Department of Computer Science
University of Maryland, College Park

IMPORTANT

- Make sure you check your e-mails every day and the messages we post on the class announcements. It is your responsibility to check them so you are aware of important information/deadlines.
- Final exam information is available on the class web page. Remembe,r the exam is on May 14, 4-6m. Check the class web page for additional information.
- Deadline for OrdersProcessor test regrades (Wed May 9). Additional information at:
- http://www.cs.umd.edu/class/spring2012/cmsc132/projects/OrdersProcessor/grading.html
- Please complete course evaluations ☺

Initialization Block

- Definition
  - Block of code used to initialize static & instance variables for class
- Motivation
  - Enable complex initializations for static variables
    - Control flow
    - Exceptions
  - Share code between multiple constructors for same class

Initialization Block Types

- Static initialization block
  - Code executed when class loaded
- Initialization block
  - Code executed when each object created
  - (at beginning of call to constructor)
- Example

```
class Foo {
    static { A = 1; }          // static initialization block
    { A = 2; }                 // initialization block
}
```

Variable Initialization

- Variables may be initialized
  - At time of declaration
  - In initialization block
  - In constructor
- Order of initialization
  1. Declaration, initialization block
     (in the same order as in the class definition)
  2. Constructor

Variable Initialization – Example

```
class Foo {
      static { A = 1; }          // static initialization block
      static int A = 2;          // static variable declaration
      static { A = 3; }          // static initialization block
      { B = 4; }                 // initialization block
      private int B = 5;         // instance variable declaration
      { B = 6; }                 // initialization block
      Foo() {                    // constructor
         A = 7;
         B = 8;
      }                          // now A = 7, B = 8
}           // initializations executed in order of number
```

BitSet Class

- Implements a set of bits where the bits of the set are indexed by nonnegative integers
- Methods
  - BitSet() – New bit set
  - BitSet(int nbits) – Bit set large enough to represent bits with indices from 0 through nbits – 1
  - and(BitSet set) – Performs logical and between the current object and the set parameter (current object is updated with the result)
  - or(BitSet set) – Performs logical or between the current object and the set parameter (current object is updated with the result)
  - cardinality() – Returns number of bits set to 1
  - flip(int bitIndex) – Sets the bit at the specified index
  - get(int bitIndex) – Returns true if the bit at bitIndex is set; false otherwise
  - length() – Index of the highest set bit + 1. It returns zero if the BitSet contains no bits set.
  - size() – Number of bits space used by the BitSet to represent bit values
  - toString() – For every bit set, the decimal representation of that index is included in the result.