CS 5044
Object-Oriented Programming with Java

Q&A Session

Final exam

- Exam will be open Friday (12/13) at noon ET through Monday (12/16) at noon ET
  - Comprehensive, with heavy emphasis on topics after the midterm
    - Parts A, B, and C are submitted via Canvas (150 points total)
      - Each part has 20 multiple choice questions (2.5 points each) with a 30-minute time limit
    - Part D is submitted via CodeWorkout (30 points total)
      - This part has 6 problems (5 points each) with a 60-minute total time limit
      - An **ungraded practice** has been published, so you can experience "exam mode"
        - » Feedback is more limited than in homework mode, plus there's a timer
      - You're expected to develop code using these techniques:
        - » Numeric arithmetic, logical, and comparison operations
        - » Branches (combinations of if, else, and else-if; possibly nested)
        - » Loops (conventional-for and enhanced-for; possibly nested)
        - » Array handling (length, get/set value, and create new; possibly nested)
        - » ArrayList handling (size, get, add, remove, and create new; possibly nested)
        - » String handling (length, equals, substring, contains, startsWith, endsWith)
        - » NOTE: Neither *recursion* nor *functional programming* should be used
  - See also Piazza @506 (Week 16 Announcements) for a few additional details
- Reminder: SPOT survey closes tomorrow (12/12) at 11:59pm ET

Course review

- Objects and Using Objects: syntax, types, variables, new objects, references
- Fundamental Data Types: primitives, arithmetic operations, String
- Implementing Classes: stack/heap, declarations, constructors, encapsulation
- Decisions and Loops: conditionals/booleans, if-else, while/for/do-while, enum
- Testing Strategies: concepts, implementation via JUnit, code coverage
- Designing Classes: mutators/accessors, side-effects, state, packages
- Arrays and ArrayList: array object, ArrayList, iteration via enhanced-for, nesting
- Java Collections: use of List/Set/Map/Stack/Queue, specifying generic types
- Inheritance and Interfaces: inheritance, abstract classes/methods, interfaces
- I/O and Exceptions: I/O streams, exception management via throw and try/catch
- Immutability and Generics: immutability, generic classes/methods, type references
- Recursion and search/sort: recursive calls, strategies/algorithms, complexity
- GUI (Swing): containers, components, layout managers, events, listeners
- Functional Programming: lambdas, method references, streams/aggregations

End-of-term updates will be posted in Piazza

Thank You!