CS 46B Spring 2017

Final Exam Notes

Rule A) Only your Green Book will be graded.

Rule B) You have to stop writing when “time” is called.

Rule C) Zero points for any solution where you call a method on a primitive.

*Copyright © 2017 by Philip Heller. No part of this document may be distributed, including to social media, without permission from the author.*

1. Do you understand Rules A, B, and C? **(1 point)**
2. INHERITANCE
   1. A class can use (“inherits”) all non-private data & methods of its superclass
   2. If a class doesn’t declare that it extends a superclass, then it extends Object
   3. Therefore all classes eventually inherit from Object
      1. toString()
      2. hashCode()
      3. equals()
   4. Constructors (“ctors”)
      1. Every class has at least 1 ctor
      2. If you don’t write any ctors, the compiler creates an invisible empty no-args ctor
      3. Every ctor begins by calling a ctor of the superclass
      4. If ctor code’s 1st line isn’t super(args list), the compiler inserts an invisible super() call (i.e. to the superclass’ no-args ctor).
      5. The ghost story
   5. Polymorphism
      1. Where a superclass is expected, a subclass is accepted
   6. Overriding
      1. Subclass provides a method with same signature as a superclass method
3. INNER CLASSES
   1. May access all variables and methods of the enclosing class, even if they are private
   2. Not a subclass of the enclosing class
      1. May extend any superclass
4. INTERFACES
   1. Lists of methods
   2. Classes declare that they implement interfaces. Compiler ensures that all methods of the interface are implemented.
5. CONVERTING & CASTING
   1. Numeric Converting & Casting
      1. Conversion
         1. Allowed when new type is wider than old type
         2. No risk of data loss
         3. Definition of “wider” – it’s not about # of bits
      2. Casting
         1. Required when new type is not wider than old type
         2. Run-time risk of data loss
         3. Cast operator tells compiler that you accept the risk
   2. Class Converting & Casting
      1. Conversion
         1. Allowed when new type is a superclass or interface of old type
         2. No risk of data loss
      2. Casting
         1. Required when conversion is not allowed
         2. Run-time risk of ClassCastException
6. Boxing / Unboxing
7. equals(), hashCode(), compareTo()
   1. Compatibility contract
   2. equals() and hashCode() are inherited from Object, but those implementations often violate the compatibility contract
   3. equals() is supposed to compute deep equality
      1. Version inherited from Object just checks shallow equality
   4. == computes shallow equality
   5. compareTo(T that) is required by the Comparable<T> interface
   6. If class X will be stored in a HashSet<X> or class X will be used as a key in a HashMap<X, Y>, then X should override hashCode() and equals()
   7. If class X will be stored in a TreeSet<X> or class X will be used as a key in a TreeMap<X, Y>, then X should override equals(), and should implement Comparable<X>
   8. Safe practice for implementing hashCode()
8. EXCEPTIONS
   1. Why exceptions are better than special return values
   2. Kinds of exception
      1. Checked
      2. Runtime
   3. How to call methods that throw checked exceptions
   4. How to call methods that throw runtime exceptions
   5. How to write methods that throw exceptions
   6. Assertions
9. INPUT/OUTPUT
   1. Streams
      1. FileReader
      2. BufferedReader
      3. PrintWriter
      4. FileWriter
   2. Many ctors and methods of these classes throw IOException
10. RECURSION
    1. Definition
    2. Examples:
       1. Triangular numbers
       2. Palindromes
       3. File lister
       4. Fibonacci
       5. Backtracking
    3. You won't have to write any recursive methods from scratch (maybe complete a recursive method that is mostly written for you)
11. ENUMS
    1. How to write one
    2. Why better than defined constants
12. SORTING & SEARCHING
    1. Sorting
       1. Selection Sort
          1. Algorithm
          2. Big-O
       2. Insertion Sort
          1. Algorithm
          2. Big-O
       3. Merge Sort
          1. Algorithm
          2. Big-O
    2. Searching
       1. Linear Search
          1. Algorithm
          2. Big-O
       2. Binary Search
          1. Algorithm
          2. Big-O
13. COLLECTIONS
    1. <Generic collections>
       1. Advantage over collecting Objects
       2. How to create one
    2. Array Lists
    3. Sets
       1. Hash sets and tree sets
       2. How they determine uniqueness
       3. Traversal order
    4. Maps
       1. Association of unique keys to values
       2. Hash maps and tree maps
       3. Traversal order of keys
       4. Traversal order of values
    5. Stacks
       1. push()
       2. pop()
    6. Custom collections
       1. Iteration
          1. Iterable interface
          2. Iterator interface
          3. Use them to allow iteration over your collection using a for-loop
          4. 2 strategies
             1. Create an inner class that manages an index
             2. Collect into an array list, then use its iterator
    7. Safe iteration
14. ABSTRACT CLASSES
15. GRAPHS
    1. Want an A? Draw diagrams when you study
    2. Nodes and edges
       1. Nodes are objects containing data
          1. Best practice: Node class is generic in the type of the data
       2. Edges are variables in node objects (next/prev/children/neighbors)
    3. Linked lists
       1. Advantage over arrays and array lists
       2. Implementation
       3. Common operations
          1. Insertion
          2. Deletion
          3. Finding
       4. Integrity
       5. Doubly linked lists
    4. Trees
       1. Terminology
       2. Implementation
       3. Binary search trees
          1. Definition
             1. Binary tree property
             2. Binary search tree property
          2. Common operations
             1. Searching
             2. Inserting
             3. Iterating
    5. General graphs
       1. Implementation
16. STRINGS
    1. Literal strings
       1. Definition
       2. The literal string pool
    2. String “addition”