Review for Midterm

Ye Yang
Stevens Institute of Technology

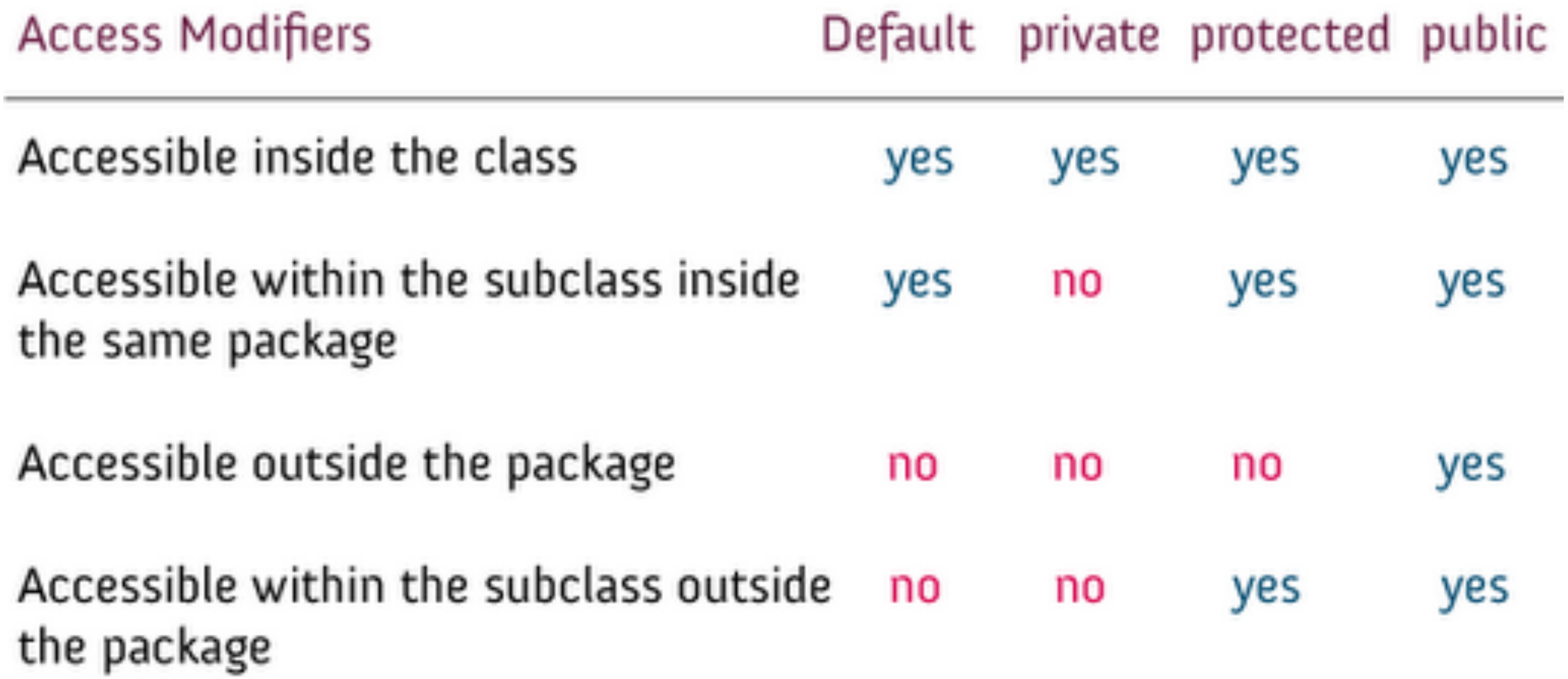
Topics Covered

- Basic Object-Oriented concepts
- Algorithm Analysis
- Lists
- Stacks and Queues

Basic Object-Oriented Concepts

- Classes and Objects
  - How to define a class?
  - How to declare an object?
  - How many objects can be declared in a program?
- Inheritance
  - Java allows single inheritance:
  - The “Object” class
  - “extends” keyword
  - Inheritance is only between classes
  - What will be inherited from super class? Both variables and methods
  - Super() method
- Data access rules
  - Common access modifiers: Default, private, protected, public (see next page for comparison)

Common Access Modifier



Access Modifiers	Default	private	protected	public
Accessible inside the class	yes	yes	yes	yes
Accessible within the subclass inside the same package	yes	no	yes	yes
Accessible outside the package	no	no	no	yes
Accessible within the subclass outside the package	no	no	yes	yes

Overloading vs. Overriding

- Overloading deals with multiple methods in the same class with the same name but different signatures
- Overloading lets you define a similar operation in different ways for different data

- Overriding deals with two methods, one in a parent class and one in a child class, that have the same signature
- Overriding lets you define a similar operation in different ways for different object types

Polymorphism

Java Collection

- Interface
  - Abstract class/method
  - “implements” keyword
  - How many interfaces can one class implement? As many as needed
- Iterator/Iterable: methods provide access to the contents of the collection
  - Object next()
  - boolean hasNext()
  - void remove()
- Comparable: used to compare the current object with the specified object
  - **public int compareTo(Object obj)**

Lists

- The List ADT, including typical List operations
- ArrayList:
  - running time of insert, remove, search at difference position in the list;
  - what to do when array is full;
- LinkedList
  - Single-linked lists vs. Double-linked lists;
  - running time of insert, remove, search at difference position in the list;
  - sentinel node (head, tail);
- Skills: Develop simple list algorithms for additional operations (swapping elements, removing duplicates, etc.)

Stacks

- Stacks
- Applications
  - Syntax Checking
  - Postfix Evaluation
- Implementation
  - Array-based implementation
  - List-based implementation

Stack Implementations

- Linked List:
  - Push(x) <-> add(x,0)
  - Pop(x) <-> remove(0)
  - Maintain “size” field appropriately
- Array:
  - Push(x) <-> Array[k++] = x
  - Pop(x) <-> return Array[--k]
  - Maintain “top” appropriately
- Skills: apply Stack operations to solve simple data manipulation problem, e.g. given two sorted stacks, merge into one sorted stack.

Queues

- Stacks are **L**ast **I**n **F**irst **O**ut
- Queues are **F**irst **I**n **F**irst **O**ut, first-come first served
- Operations: enqueue/dequeue, or add/remove, or offer/pull
- Implementations
  - Linked List
    - add(x,0) to enqueue, remove(N-1) to dequeue
  - Circular Array
    - Array List won’t work well: since add(x,0) is expensive

Circular Array Queue

- Don’t bother shifting after removing from array list
- Keep track of start and end of queue
- When run out of space, wrap around
  - modular arithmetic
- When array is full, increase size using list tactic
- Skills: use queue operations on a queue to find min, max, and maintain the queue intact. Assume no additional data structure for storage.

Algorithm Analysis

- Time vs. Space analysis
- Big-O notation
  - What is the implication of Big-O representation?
  - Be able to compare growth of functions using big-O notation.
  - Given an algorithm (written in Java), estimate the asymptotic run time (including nested loops and simple recursive calls).
  - Given a problem, choose appropriate data structure to meet desired run time constraint.

Data Structure Performance Comparison

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)