Software Development and Abstract Data Types

Chapters 2, 4, 10
Phases of Software Development

1. Problem statement
2. Specification of the task
3. Design of a solution for each task
4. Implementation of a solution
5. Testing and Debugging
6. Documentation and Maintenance
Specification of a Method

- Short introduction/summary
- Description of parameters
- Preconditions
- Postconditions or Returns
- Exceptions thrown
- Special notes on usage
Example

area
public static double area(double radius)

Parameters:
  radius – radius of a circle in inches

Preconditions:
  radius > 0

Returns:
  Returns the area of the circle with the given radius.

Throws: IllegalRadius
  Indicates that the radius is non positive.
Efficiency

• What’s the best way to program an algorithm?

• Efficiency in terms of
  – Time (running time)
  – Space (memory requirements)
  – Resources (Input/Output such as disk I/O)

• Most analysis focuses on time efficiency
Order of Complexity

• Measuring running time (Benchmark, Analysis)
• Count the number of “operations”
• What is an “operation”?  
• Example:  
  \( C = A + B; \)
• 1 operation?  
• 4 operations?  
  (LOAD A, LOAD B, ADD, STORE C)
Order of Complexity (cont’d)

for (x=1; x<=n; x++)
    c = a + b;
# of ops: n 4n 3n+2 ?
for (x=1; x<=n; x++)
{
    c = a + b;
    d = e + f;
}
# of ops: 2n 8n 4n+2 ?
Big O Notation (upper bound)

• Given an algorithm/routine processing n inputs, what is the number of operations as a function of n?
• Big O expresses this function as a simplified function of n (input size)
• \( O(n) \) – a function of n
• \( O(n^2) \) – a function of \( n^2 \)
Big O Notation (cont’d)

• Let \( T(n) \) and \( f(n) \) be functions mapping nonnegative integers to real numbers.

• We say that \( T(n) \) is \( O(f(n)) \) if there is a real constant \( c > 0 \) and an integer constant \( n_0 \geq 1 \) such that \( T(n) \leq cf(n) \) for every integer \( n \geq n_0 \).
Big O Examples

for (x=1; x<=n; x++)
    c = a + b;

n operations
O(n)

for (x=1; x<=n; x++)
{
    c = a + b;
    d = e + f;
}

2n operations
O(2n) = O(n)
Big O Examples (cont’d)

```c
for (x=1; x<=n; x++)
    c = a + b;

for (x=1; x<=n; x=x*2)
    c = a + b;
```

- **n+1 operations**
  - $O(n+1) = O(n)$

- **Assume n is a power of 2 for simplicity.**
  - $\log_2 n + 1$ operations
  - $O(\log_2 n + 1) = O(\log n)$
Big O Examples (cont’d)

```c
for (x=1; x<=n; x++) {
    d = e + f;
    for (y=n; y>0; y--) {
        c = a + b;
        c = c * 2;
        d = c + d;
    }
}
```

$3n^2 + n$ operations

$O(3n^2 + n) = O(n^2)$
Worst-, Average-, Best-Case

- Many algorithms give their worst-case performance in terms of big-O.
- Other measures include average-case or best-case analysis.
- Example: Sequential Search on n elements

  *This time, an operation is a “comparison”.*
  
  - Worst Case: \( O(n) \)
  - Average Case: \( O(n/2) = O(n) \)
  - Best Case: \( O(1) \)
One more example

• Consider an algorithm that processes $n$ data items in one minute. How long will it take to process 32 times as many items on the same computer using the same algorithm?

$$\text{original_time} \times \frac{\text{new_number_of_ops}}{\text{original_number_of_ops}}$$

• # of operations \hspace{1cm} APPROX TIME
  
  \begin{itemize}
    \item $n = O(n)$
    \item $n^2 = O(n^2)$
    \item $2^n = O(2^n)$
  \end{itemize}
Another view

• What is the maximum number of inputs that can be processed by an algorithm in one hour if one operation takes 1 microsecond?

<table>
<thead>
<tr>
<th>Number of Ops</th>
<th>Max. Problem Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>– 400n = O(n)</td>
<td>n =</td>
</tr>
<tr>
<td>– 2n^2 = O(n^2)</td>
<td>n =</td>
</tr>
<tr>
<td>– 2^n = O(2^n)</td>
<td>n =</td>
</tr>
</tbody>
</table>
Misuse of the Big O!

- Number of Operations: $10^{10}n$
  Is this $O(n)$? (probably not)
- What’s considered an efficient measure for an algorithm?
  Yes: $O(\log n)$, $O(n)$, $O(n \log n)$, even $O(n^2)$ sometimes
  No: $O(n^{100})$, $O(2^n)$
- Is $O(1)+O(1)+ \ldots + O(1) = O(1)$?
Tale of Two Algorithms

• Given an array X storing n numbers, we want to compute an array A such that A[i] is the average of elements X[0],…,X[i], for i=0,…,n-1.
• Calculating “prefix averages”
• Used in economics: Given year-by-year returns on a mutual fund, an investor will want to know the average return over the past 3 years, 5 years, 10 years, etc.
First Attempt

`prefixAverages1(X)`

Let `A[ ]` be an array of `n` numbers

for `i ← 0` to `n-1` do

    `sum ← 0`

    for `j ← 0` to `i` do

        `sum ← sum + X[j]`

    `A[i] ← sum/(i + 1)`

return array `A`

Order of Complexity?

CAREFUL!
An Observation

• In the original algorithm, a lot of computations are recomputed over and over again.
• \( A[i-1] = (X[0] + X[1] + \ldots + X[i-1]) / i \)
• \( A[i] = (X[0] + X[1] + \ldots + X[i-1] + X[i]) / (i+1) \)
• Let the prefix sum
  \( S_i = X[0] + X[1] + \ldots + X[i-1] + X[i] \)
• \( A[i-1] = (S_{i-1}) / i \)
• \( A[i] = (S_{i-1} + X[i]) / (i+1) \)
prefixAverages2(X)

Let A[ ] be an array of n numbers
sum ← 0
for i ← 0 to n-1 do
    sum ← sum + X[i]
    A[i] ← sum/(i + 1)
return array A

Order of Complexity?
Competing Algorithms

- Algorithm A: $O(n \log n)$
- Algorithm B: $O(n^2)$
- Which is more efficient?
Abstract Data Types

• Information hiding – separation of specification from implementation

• Abstract Data Type –
  A mathematical specification of a data structure that specifies:
  – the type of data stored
  – the operations supported on that data
  – the types of parameters of the operations

• An ADT specifies what, but not how
Example 1: Location ADT
Constructor

```
public Location(double xInitial, double yInitial)
```

Construct a location with specified coordinates.

Parameters:
- `xInitial` – the initial x coordinate of this Location
- `yInitial` – the initial y coordinate of this Location

Postcondition:
This Location has been initialized at the given coordinates.
clone

public Object clone()

Generate a copy of this Location.

Returns:

The return value is a copy of this Location.

Special note:

The return value must be typecast to a Location before it can be used.
public static double distance(Location p1, Location p2)

Compute the distance between two Locations.

Parameters:
- p1 – the first Location
- p2 – the second Location

Returns: The distance between p1 and p2.

Special Note: Returns Double.NaN if either Location is null.
equals

public boolean equals (Object obj)

Compare this Location to another for equality.

Parameters:
 obj – an object with which this Location is compared

Returns: True if obj refers to the same Location as this Location object. False otherwise.

Special Note: Returns false if obj is null or not a Location object.
Specification (cont’d)

getX (and getY)

```java
public double getX() ( or getY() )
```

Get the x (or y) coordinate of this Location

Returns: the x (or y) coordinate of this location
public static Location midpoint (Location p1, Location p2)

Generates and returns a Location halfway between two Locations

Parameters:
  p1 – the first Location
  p2 – the second Location

Returns: a Location halfway between p1 and p2

Special note: Returns null if either p1 or p2 is null.
rotate90

public void rotate90()
Rotate this Location 90 degrees in a clockwise fashion around the origin

Postcondition: This Location has been rotated clockwise 90 degrees around the origin
shift
public void shift(double xAmount,
   double yAmount)
Move this location by given amounts along the x and y axes.
Postcondition: This location has been moved by the given amounts along the two axes.
Special note: (see text)
Specification (cont’d)

toString

public String toString()

Generate a string representation of this Location object.

Returns: a string representation of this Location
Method Types

• Accessor method – returns information about the state of an object without altering the state of the object [getX, getY]
• Modification (mutator) method – may change the state of an object through its invocation [rotate90, shift]
• Static method – returns information about a set of one or more objects [distance, midpoint]
• Support method – provides common support for objects [the constructor, clone, equals, toString]
public class Location implements Cloneable {

    private double x;  // state variables
    private double y;

    public Location (double xInitial, double yInitial) // constructor
    {
        x = xInitial;
        y = yInitial;
    }
}
Implementation (cont’d)

```java
public Object clone()
{
    Location answer;
    try {
        answer = (Location)super.clone();
    }
    catch (CloneNotSupportedException e)
    {
        throw new RuntimeException("...");
    }
    return answer;
}
```
public static double distance (Location p1, Location p2) {
    double a, b, c_squared;
    if ((p1==null)||(p2==null))
        return Double.NaN;
    a = p1.x - p2.x;
    b = p1.y - p2.y;
    c_squared = a*a + b*b;
    return Math.sqrt(c_squared);
}
Implementation (cont’d)

```java
public boolean equals(Object obj)
{
    if (obj instanceof Location)
    {
        Location candidate = (Location) obj;
        return (candidate.x == x) &&
                (candidate.y == y);
    }
    else
        return false;
}
```
Implementation (cont’d)

```java
public double getX()
{
    return x;
}
public double getY()
{
    return y;
}
```
public static Location midpoint (Location p1, Location p2) {
    double xMid, yMid;
    if ((p1==null) || (p2==null))
        return null;
    xMid = (p1.x/2)+(p2.x/2);
    yMid = (p1.y/2)+(p2.y/2);
    Location answer =
        new Location(xMid, yMid);
    return answer;
}
Implementation (cont’d)

```java
public void rotate90() {
    double xNew, yNew;
    xNew = y;
    yNew = -x;
    x = xNew;
    y = yNew;
}
```
Implementation (cont’d)

```java
public void shift(double xAmount,
                   double yAmount) {
    x += xAmount;
    y += yAmount;
}

public String toString() {
    return "(x=" + x + "  y=" + y + ")";
}

} // end class Location
```
Use of Location ADT

class LocationTester {
    public static void main(String[] args) {
        Location server = new Location(2.0, 4.5);
        Location mobile = (Location) server.clone();
        mobile.shift(-3.0, -3.0);
        System.out.println("The devices are " +
                Location.distance(server, mobile) +
                " blocks away from each other.");
    }
}

etc.
Example 2: Bag ADT

- Bag: A collection of items of the same type.
- A specific item may appear any number of times in a bag.
- A bag is not a set.
- A bag is not ordered.
- The items of a bag will be stored in an array (for now).
- A bag may have limited size due to memory constraints.
Constructors

public IntArrayBag()
public IntArrayBag(int initialCapacity)

Construct a bag of integers with capacity 10 (default) or initialCapacity.

Precondition: initialCapacity ≥ 0

Postcondition: This bag is empty and has an initial capacity.

Throws: IllegalArgumentException, OutOfMemoryError
getCapacity
public int getCapacity()
Determines the current capacity of this bag.
Returns: the current capacity of this bag.

size
public int size()
Determines the number of elements in this bag.
Returns: the number of elements in this bag.
ensureCapacity

public void ensureCapacity
   (int minimumCapacity)

Change the current capacity of this bag.
Parameters: minimumCapacity – the new capacity for this bag
Postcondition: This bag’s capacity has been changed to minimumCapacity, if this is greater than its current capacity.
Throws: OutOfMemoryException
add

public void add(int element)
Add a new element to this bag.
Parameters:
   element – the new element being added to the bag
Postcondition: A new copy of the element has been added to this bag.
Special Note: If the new element cannot be stored in the bag at its current capacity, the bag’s capacity is increased.
Throws: OutOfMemoryError
Specification (cont’d)

addAll

public void addAll(IntArrayBag addend)
Add the contents of another bag to this bag.
Parameters:
  addend – a bag whose contents will be added to this bag
Postcondition: This bag will contain its original contents and the contents of the other bag.
Throws: NullPointerException, OutOfMemoryError
union

public static IntArrayBag union
    (IntArrayBag b1, IntArrayBag b2)

Create a new bag that contains all the elements from two other bags.

Parameters:
    b1 – the first of two bags
    b2 – the second of two bags

Precondition: neither b1 nor b2 is null

Returns: A new bag that is the union of b1 and b2

Throws: NullPointerException, OutOfMemoryError
countOccurrences

public int countOccurrences(int target)
Count the number of occurrences of a particular value in this bag.

Parameters:
  target – the element that needs to be counted

Returns: The number of times the target is in this bag.
remove
public boolean remove(int target)
Remove one copy of a specified element from this bag.
Parameter:
    target – the element search for in this bag for removal
Postcondition: If the target was found in this bag, then one copy is removed and the method returns true. Otherwise, this bag remains unchanged and the method returns false.
trimToSize

public void trimToSize()
Reduce the current capacity of this bag to its actual size.
Postcondition: This bag’s capacity has been changed to its current size.
Throws: OutOfMemoryError
clone

public Object clone()

Generate a copy of this bag.

Returns:
The return value is a copy of this bag.

Special note:
The return value must be typecast to an IntArrayBag before it can be used.

Throws: OutOfMemoryError
Invariant of the ADT

• An invariant is a condition that remains true before and after some operation is performed (i.e. precondition = postcondition)
• All the methods of an ADT (except the constructors) must ensure that the invariant of the ADT is valid before and after execution.
Invariant of the Bag ADT

• Let data = the array that holds the bag items
• Let data.length = the capacity of the array
• Let manyItems = the number of items in the bag (i.e. its size)
• INVARIANTS:
  – The elements of a bag are stored in data[0..manyItems-1].
  – manyItems ≤ data.length
public class IntArrayBag
  implements Cloneable
{
  private int[] data;
  private int manyItems;
}
Implementation (cont’d)

```java
public IntArrayBag(int initialCapacity) {
    if (initialCapacity < 0)
        throw new IllegalArgumentException();
    manyItems = 0;
    data = new int[initialCapacity];
}

public IntArrayBag() {
    manyItems = 0;
    data = new int[10];
}
```
Implementation (cont’d)

```java
public int getCapacity() {
    return data.length;
}

public int size() {
    return manyItems;
}
```

Order of Complexity?
Sidenote: System.arraycopy

System.arraycopy(src, si, dest, di, n);
src = reference of array to copy FROM
si = starting position to copy FROM
dest = reference of array to copy TO
di = starting position to copy TO
n = how many elements to copy
Implementation (cont’d)

```java
public void ensureCapacity(int minimumCapacity) {
    int biggerArray[];
    if (data.length < minimumCapacity) {
        biggerArray = new int[minimumCapacity];
        System.arraycopy(data, 0,
                         biggerArray, 0, manyItems);
        data = biggerArray;  // previous data?
    }
}
} Order of Complexity?
Implementation (cont’d)

```java
public void add(int element) {
    if (manyItems == data.length)
        ensureCapacity(manyItems*2+1);
    data[manyItems] = element;
    manyItems++;
}
```

Order of Complexity?
Implementation (cont’d)

```java
public void addAll(IntArrayBag addend) {
    ensureCapacity(manyItems +
                   addend.manyItems);
    System.arraycopy(addend.data, 0, data,
                     manyItems, addend.manyItems);
    manyItems += addend.manyItems;
}
```

Order of Complexity?
public static IntArrayBag union
    (IntArrayBag b1, IntArrayBag b2) {
    IntArrayBag answer = new IntArrayBag
        (b1.getCapacity() + b2.getCapacity());
    System.arraycopy(b1.data, 0, answer.data,
        0, b1.manyItems);
    System.arraycopy(b2.data, 0, answer.data,
        b1.manyItems, b2.manyItems);
    answer.manyItems =
        b1.manyItems + b2.manyItems;
    return answer;
}

Order of Complexity?
Implementation (cont’d)

```java
public int countOccurrences(int target) {
    int answer = 0;
    int index;
    for (index = 0; index < manyItems; index++) {
        if (target == data[index])
            answer++;
    }
    return answer;
}

Order of Complexity?
Implementation (cont’d)

public boolean remove(int target) {
    int index = 0;
    while ((index < manyItems) &&
        (target != data[index]))
        index++;
    if (index == manyItems)
        return false;
    else {
        manyItems--;  
data[index] = data[manyItems];
        return true;
    }
}  

Order of Complexity?
Implementation (cont’d)

```java
public void trimToSize() {
    int trimmedArray[];
    if (data.length != manyItems) {
        trimmedArray = new int[manyItems];
        System.arraycopy(data, 0,
                         trimmedArray, 0, manyItems);
        data = trimmedArray; // previous data?
    }
}
```

Order of Complexity?
Implementation (cont’d)

```java
public Object clone() {
    IntArrayBag answer;
    try {
        answer = (IntArrayBag) super.clone();
    }
    catch (CloneNotSupportedException e) {
        throw new RunTimeException("...");
    }
    answer.data = (int []) data.clone();
    return answer;
}
} // end class IntArrayBag
```

Order of Complexity?
Order of Complexity

- Given a bag with n items and capacity c.
- **add**
  - Without a capacity increase: \( O(1) \)
  - With a capacity increase: \( O(n) \)
- **countOccurrences** \( O(n) \)
- **getCapacity** \( O(1) \)
- **remove** \( O(n) \)
- **clone** \( O(c) \)