Week 7 worksheet: Complexity Analysis and Generics

Total points: 10

Out: February 18 (Tuesday)

Due: February 21 (Friday end of day [2359 CDT according to D2L])

## What to submit?

Upload only one Word or PDF file to the designated D2L folder.

## Running time assumptions

* Assume that basic arithmetic operations (+, -, \*, /, %) run in constant time.
* Assume that the System.out.println( … ) statement runs in constant time.

# Exercise 1: Runtime analysis

[1 pt each] What are the big-O running times of the following methods as a function of the input parameter(s)? In each case, the correct answer will be one of the following:

O(1), O(log n), O(n), O(n log n), O(n2), O(2n), O(n!).

public void method1( int n ) {

for ( int j = 1; j <= n; j++ ) {

for ( int k = 1; k < n; k++ ) {

System.out.println ( j \* k ) ;

}

}

}

Big-O running time of method1 is O(n^2)

// Assume the method Math.max(…) runs in constant time.

public void method2( int n ) {

for ( int j = n; j <= 2; j /= 2 ) {

for ( int k = 0; k < j; k++ ) {

System.out.print( Math.max( j, k ) );

}

}

}

Big-O running time of method2 is O(n)

public void method3( int n ) {

for ( int j = 1; j <= n; j \*= 2 ) {

System.out.println ( j \* k ) ;

}

}

Big-O running time of method3 is O(log n)

public void method4( int n ) {

for ( int j = 1; j <= n; j++ ) {

System.out.println ( j ) ;

}

}

Big-O running time of method4 is O(n)

# Exercise 2: Multivariable runtime analysis

[1 pt each] What are the big-O running times of the following methods as a function of the input parameters?

public void method5( int m, int n ) {

for ( int j = 1; j <= m; j++ ) {

for ( int k = 1; k < n; k++ ) {

System.out.println ( j \* k ) ;

}

}

}

Big-O running time of method5 is O(N log N)

public void method6( int m, int n ) {

for ( int j = 1; j <= m; j \*= 2 ) {

for ( int k = 1; k < n; k \*= 2 ) {

System.out.println ( j \* k ) ;

}

}

}

Big-O running time of method6 is O(log N)

# Exercise 4: Linked list removeSecond

[4 pts] Suppose that you have a linked list of Nodes of Thing objects. It has two named nodes, “head” and “tail”. “tail” has a null “next” reference.

A sample such list with Strings would look like this:

public class ThingNode {

Thing data;

ThingNode next;

public Thing getData() { return data; }

public ThingNode getNext() { return next; }

public void setData( E data ) { this.data = data; }

public void setNext ( ThingNode next ) { this.next = next; }

}

Write a method removeSecond that removes the second node of the linked list then reconnects the list. If there is zero or one element, return null. The signature of this method is

public Thing removeSecond(Thing target, ThingNode head){

if (head == null || head.getNext() == null){

return null;

}

ThingNode first = head;

ThingNode second = head.getNext();

ThingNode third = second.getNext();

Thing removed = second.getData();

first.setNext(third);

second.setNext(null);

return removed;

}