**MIDTERM STUDY GUIDE**

The mid-term is based on Lectures 1 – 12, but will not include questions about Java GUIs.

The mid-term is closed book. Do not log into your computers. No calculators, iPods, iPads or notes.

The mid-term will be a mix of questions below and questions that you have not seen before.

Schedule:

Mon 10/2/2017 (morning) – lecture #10-11

Wed 10/4/2017 (morning) – lecture #12

Wed 10/4/2017 (lab; afternoon) – study session; work on study guide

Mon 10/9/2017 – fall break – no class

Wed 10/11/2017 - (morning) – midterm at 9:30 am

Wed 10/11/2017 – Our first GUI/threading lab

Lab exercises (to get ready for the mid-term; no need to hand this in).

implement comparable() in your FastaSequence class so that FastaSequence has a natural order defined by sequence length (with the shortest sequences at top).

In addition, make Comparator<FastaSequence> to allow a List<FastaSequence> to be sorted by:

An alphabetical sort of the sequence (so “AAA” would be sorted ahead of “AAC”)

An alphabetical sort of the headers

By the GC content (for a DNA sequence)

By the number of valid characters (“A”, “C”, “G”, “T”) in each sequence (with the sequence with the fewest number of valid characters at the top of the list).

Call Collections.sort(...) on your ArrayList to see if all of your sorting code worked.

(1) Within an if statement what is the difference between || and &&

(2) Write a function called “numGCs” that takes as a parameter a String and returns an integer that is the number of times G and C is observed within the string

(3) Write a function “bothPositive” that takes two integers and returns an integer that is 1 if both integers are greater than 0, otherwise returns 0

(4) Write a function called skipEveryOther(String s) that prints out every other character in a string to the console.

(5) If x is an integer, what is the difference between x++ and ++x

(6) What does this code do?

**public** **void** swap(**float** f1, **float** f2)

{

**float** temp = f1;

f1 = f2;

f2 = temp;

}

(7) What is the difference between an Abstract class and an interface? When should you use one vs. the other?

(8) True or false: The code in a protected constructor in an abstract class can never be executed. Why or why not?

True of false: The code in a private constructor in an abstract class can never be executed. Why or why not?

(9) Given this code:

**public** **abstract** **class** Shape

{

**abstract** **public** **double** getArea();

**abstract** **public** **double** getShapeName();

}

**public** **class** Circle **extends** Shape

{

**private** **final** **double** radius;

**public** **double** getRadius()

{

**return** **this**.radius;

}

**public** Circle(**double** radius)

{

**this**.radius = radius;

}

@Override

**public** **double** getArea()

{

**return** Math.PI \* radius \* radius;

}

@Override

**public** String getShapeName()

{

**return** "Circle";

}

}

(A) The following does not compile. Why?

**public** **static** **void** main(String[] args)

{

Shape shape = **new** Shape();

}

(B) What are the advantages and disadvantages of making the instance variable radius final in the Circle class? What are the advantages and disadvantages of making it private?

(C) What does the “@Override” do?

(D) The following does not compile (the error is in the second line). Why?

**public** **static** **void** main(String[] args)

{

Shape shape = **new** Circle(5);

System.out.println(shape.getRadius());

}

(E) The following code works fine. Why does the code compile even though the constructor for Circle takes a double and not an integer?

**public** **static** **void** main(String[] args)

{

**int** aRadius = 5;

Shape shape = **new** Circle(aRadius);

System.out.println(shape.getShapeName());

}

(F) Change the code above so that Shape is an interface. Make Circle implement this new interface Shape.

(10) Given the following code;

**public** **static** **void** main(String[] args) **throws** Exception

{

**long** startTime = System.currentTimeMillis();

String s="";

**for**( **int** x=0; x < 10000; x++)

s += x;

**float** numSeconds = (System.currentTimeMillis() - startTime) / 1000f;

System.out.println( numSeconds + " seconds" );

System.out.println(s);

}

(A) This code runs very slowly. How can it be speeded up? Change it so that it runs faster.

(B) In the calculation of numSeconds, why is the difference in milliseconds divided by 1000f and not just 1000 ?

(11) Given this:

List<String> list = **new** ArrayList<String>();

// code that adds stuff to the list

Describe at least two ways to iterate through the list.

What is the advantage of having list be of type List and not of type ArrayList? How would the performance change if the list were instead a LinkedList?

/\*

\* Reverses the order of the array. That is, for an array of size n, the 0th element becomes the n-1th element, the 1st element becomes the n-2th element and so forth

\*/

**public** **static** **void** reverseArray(**float**[] a)

{

}

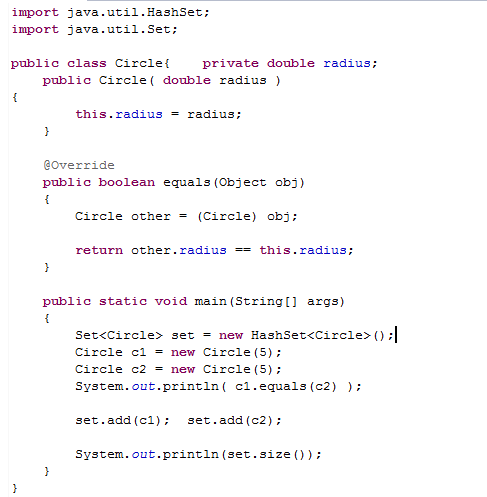
(12) Implement the above method

(13) A Set is not allowed to have two equal elements. The below code, however, prints out

true

2

Why? Fix the code so that the HashSet works properly.



(14) Here is a program to print out one dollar minus 9 dimes

**public** **static** **void** main(String[] args)

{

System.out.println( 1.00 - 9 \* .10);

}

It prints out 0.09999999999999998 not 0.10? Why? Write two different versions of the program that make the same calculation but print out the correct answer.

(15) Implement this method:

/\*Returns a List containing all the Integers in inList that are even. The order of Integers in the return List is the same as in inList\*/

**public** **static** List<Integer> getEvenNumbers( List<Integer> inList )

{

}

(16) How is a private method different from a protected method, a public method or a method with default visibility?

(17) What is the difference between ArrayList and LinkedList? What is the performance time of basic operations for ArrayList and LinkedList?

(18) This code doesn’t compile. Why?

**public** **int** aMethod(**int** anInt)

{

**return** anInt;

}

**public** **static** **void** main(String[] args)

{

System.out.println( aMethod(5) );

}

(19) Why does this code fail to compile?

(20)What is the difference between .equals() and ==. What does .equals() do on the Object class?

------------------------------------------------

(21) In your own words, how do HashSets achieve constant time for basic operations? (Also,

what is constant time)

---------------------------------------------------------------

(22) What does the method .hashCode() in object do. When you over-ride it, what rules should

you follow.

--------------------------------------------------------------------

(23)Does this implementation of HashCode obey the contract for .hashCode(). Why or why not?

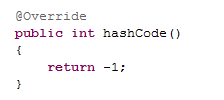
If so,



If so, is it efficient? Why or why not?

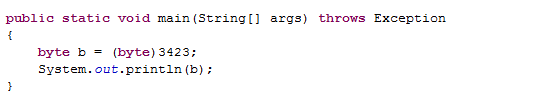
(24) Does this implementation of HashCode obey the contract for .hashCode(). Why or why

not? If so, is it efficient? Why or why not?

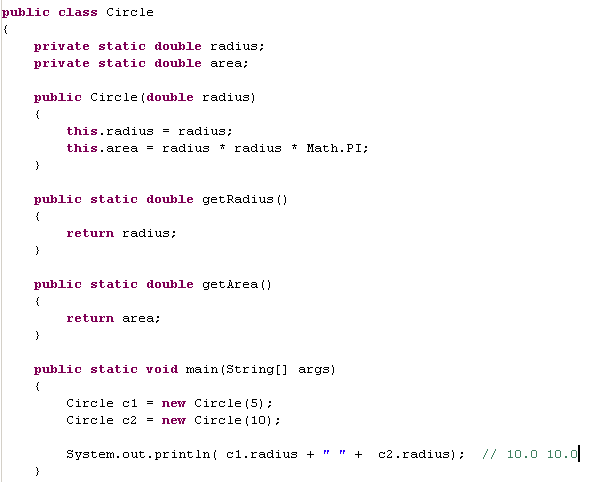


(25) Why do you need to override hashcode() when you override equals(). What are the consequences of failing to do that?

(26) This code prints out 95 and not 3423. Why?

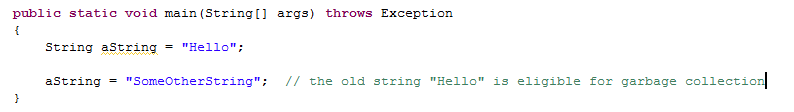


(27) This code compiles but violates instance encapsulation. Why? What is the difference between static and non-static data? What is the difference between static and non-static methods?



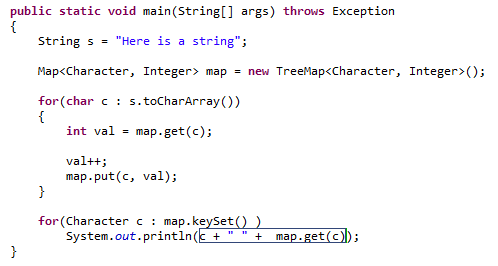
(28) What is the difference between a constructor and a static initializer?

(29) Strings are immutable in Java but in the below code, the reference appears to change. How is this possible. Which string literal is eligible for garbage collection after the 2nd line has finished executing?

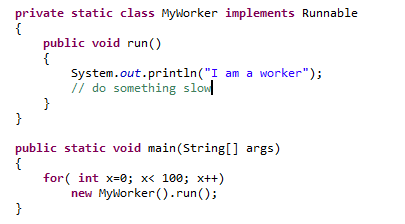


(30) When should you override .toString()? When should you override .equals()? What is the contract for overriding .equals (i.e. what do you need to ensure when you override .equals())?

-

(31) This code is designed to count the number of times each character occurs in a string. It throws a null pointer exception. Why? Modify the code so that it works correctly. 

1. What is the difference between implementing interface Runnable and extending class Thread. How do you start threads in both cases? Write an example of starting multiple threads in both cases.
2. How many threads are run in parallel in the following code:



How would you modify the code to make it run more threads in parallel?

1. What is the difference between a LinkedHashMap, a TreeMap and a HashMap?
2. What is the average performance characteristics of get and set operations on a LinkedList, ArrayList and a HashMap? Which of these data structures allow duplicate elements?