Question 1. (4 points) Fill in the blanks to show the values of the given expressions immediately after this code executes.

```
double rate = 10.0;
int entries = 3;
double[] weight = new double[entries];
for (int i=0; i<weight.length; i++) {
    weight[i] = rate*i;
}
boolean ifPositive = (weight[1] > weight[0]);
double span = weight[weight.length-1] - weight[0];
```

a. weight[2] ______**20.0**___

b. weight.length ______**3**______

c. ifPositive ______**true**___

d. span ______**20.0**___

Question 2. (4 points) Circle T or F to indicate if the following statements are true or false.

a. [✓ T] F A class can implement one or more interfaces.

b. [✓ T] F Several similar objects can be created from one class definition using the `new` operator one or more times

c. T [✓ F] The "==" operator may return an integer result, depending on the values involved.

d. [✓ T] F When the `String` method "`public boolean equals(Object anObject)`" returns, its result is always a `boolean` value.

Question 3. (3 points) Read the following method from the sim.BasicMatrixModel class in project 3.

```
private int colWrap(int col) {
  if (col >= colCount) {
    return 0;
  } else if (col < 0) {
    return colCount-1;
  } else {
    return col;
  }
}
```

a. Is this method a static method? Yes [✓ No]

b. What is the type of the value returned by this method? **int**

c. Variable `colCount` is one of the following: parameter, local variable, or instance variable. Which one is it? **instance variable**

Question 4. (6 points) Consider the following simple Tree class that models the trees in a forest.

a. How many constructors are defined for this class? _______1______

b. How many instance variables are defined for this class? _______2______

c. The Tree constructor takes arguments of type `double`. If you call this constructor with `int` values (for example, "`new Tree(10,20)`" will it compile and work correctly? _______yes____

d. Write the code for method "`public double getGrowthRate()`" as described in the comments.

```
/** This class describes a very simple model of a living tree. */
public class Tree {
  private double height;
  private double age;
  /**
   * Initialize a new Tree.
   * @param meters height of the tree in meters
   * @param years age of the tree in years
   */
  public Tree(double meters, double years) {
    height = meters;
    age = years;
  }
  /** @return the current height of this Tree in meters. */
  public double getHeight() {
    return height;
  }
  /** @return the current age of this Tree in years. */
  public double getAge() {
    return age;
  }
  /** @param m the new height of this Tree in meters. */
  public void setHeight(double m) {
    height = m;
  }
  /** @param y the new age of this Tree in years. */
  public void setAge(double y) {
    age = y;
  }
  /**
   * Calculate the rate of growth in meters/year, based on the current height
   * and the current age. If age is less than or equal to zero, return 0.0,
   * otherwise return height divided by age.
   * @return the rate of growth in meters/year
   */
  public double getGrowthRate() {

      if (age <= 0.0) {

         return 0.0;

      } else {

         return height/age;

      }

  }
}
```

Question 5. (10 points) Consider the following simple `Forest` class that deals with a collection of `Trees`. The `Tree` class is the one defined in the previous problem.

a. Implement the `Forest` constructor according to the comments.

b. Implement the `findOneTallTree(double h)` method according to the comments.

```
import java.util.*;
/** This class models a collection of Trees. */
public class Forest {
  private ArrayList woods;         // collection of trees in this Forest
  /**
   * Initialize a new Forest that has the given number of new
   * Trees, all of the given height and age.
   * @param treeCount the number of Trees
   * @param treeHeight the size of all the original Trees
   * @param treeAge the age of all the original Trees
   */
  public Forest(int treeCount, double treeHeight, double treeAge) {

      words = new ArrayList();
      for (int k = 0; k < treeCount; k++) {
         woods.add(new Tree(treeHeight, treeAge));
      }
      // The solution could have used addTree(...) instead of woods.add(...)
  }
  /**
   * Add a Tree to the Forest.
   * @param t the Tree to add
   */
  public void addTree(Tree t) {
    woods.add(t);
  }
  /**
   * Find one Tree whose height is greater than the given value.
   * @param h find a Tree with height greater than this value
   * @return null or a reference to a Tree taller than h
   */
  public Tree findOneTallTree(double h) {

      Iterator it = woods.iterator();
      while (it.hasNext()) {
         Tree t = (Tree)it.next();
         if (t.getHeight() > h) {
            return t;
         }
      }
      return null;
      // A solution with a for loop using get(k) to access the trees would also work
  }
}
```

Question 6. (2 points) Consider the following two classes. Assume that they are in files `Divider.java` and `RunDivider.java` and that they compile correctly.

```
public class Divider {
  public int splitCount(int n) {
    int count = 0;
    int limit = n;
    while (limit > 1) {
      limit = limit / 2;
      count++;
    }
    return count;
  }
}

public class RunDivider {
  public static void main(String[] arg) {
    Divider d = new Divider();
    int answer = d.splitCount(8);
  }
}
```

a. Is variable "`d`" in `RunDivider` a local variable or an instance variable? ______**local**__

b. This program can be run with the statement "`java RunDivider`". What is the value of variable "`answer`" just before the main method exits? ______**3**_____

Question 7. (1 point) Consider the following class definition in file `Stringer.java`.

```
public class Stringer {
}
```

We can use this class from the DrJava interactions pane like this:

```
> Stringer obj = new Stringer();
> String s = obj.toString();
> System.out.println(s);
Stringer@3a9bba
>
```

The method "`String toString()`" is not defined in class `Stringer` and yet this code operates correctly. What is the name of the class that defines the method "`String toString()`" that was executed in this example? ______**Object**______

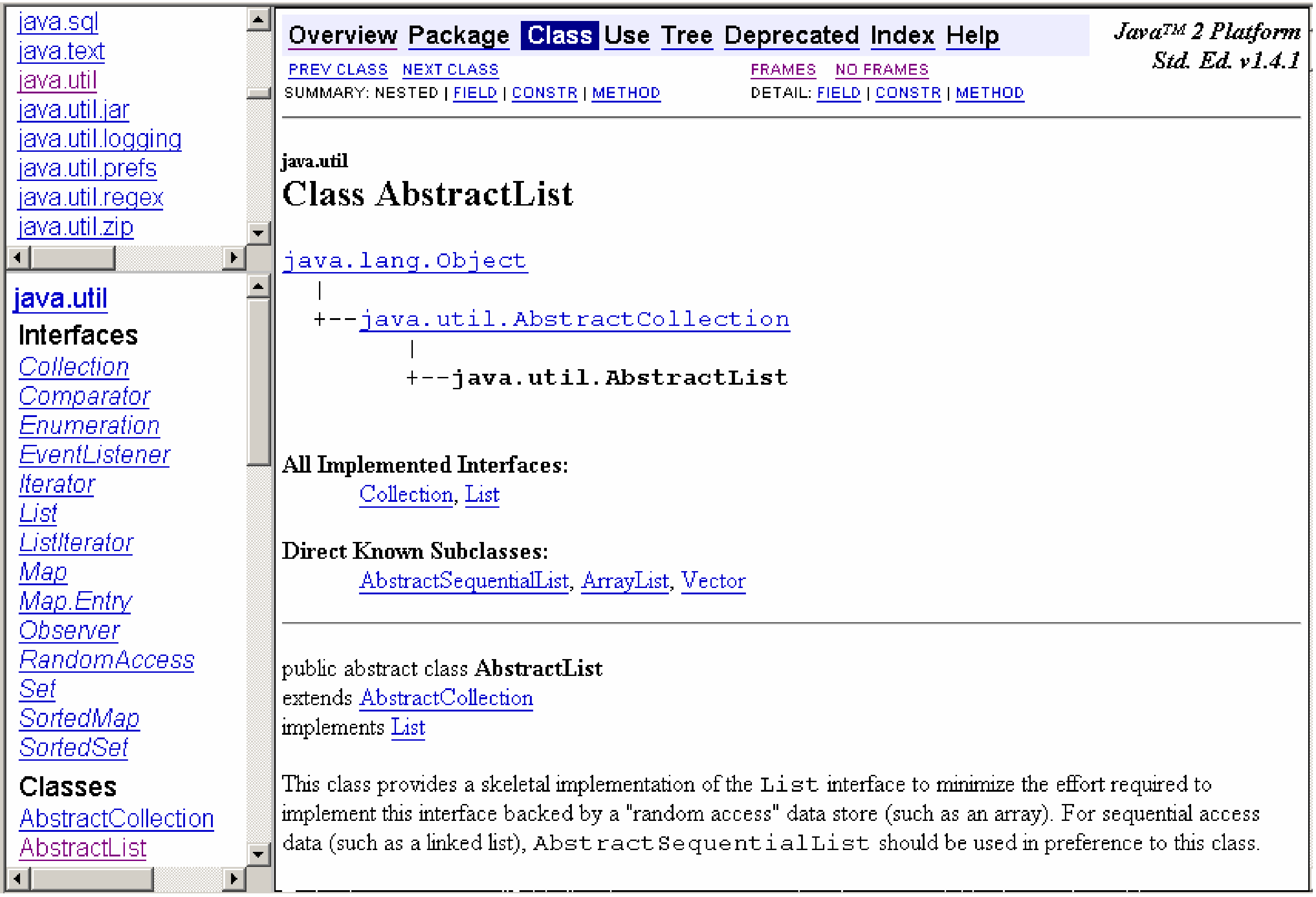
Question 8. (4 points) Given the page of documentation shown decide if the following statements are true or false.

a. [✓ T] F You can store a reference to an object of type `Vector` in a variable of type `AbstractList`.

b. [✓ T] F `AbstractCollection` is one of the superclasses of `ArrayList`.

c. T [✓ F] There is an interface named `AWTEventListener` defined in the `java.util` package.

d. [✓ T] F `AbstractSequentialList` implements the `Collection` interface.



java.util

Class AbstractList

```
java.lang.Object
  |
  +--java.util.AbstractCollection
        |
        +--java.util.AbstractList
```

All Implemented Interfaces:

Collection, List

Direct Known Subclasses:

AbstractSequentialList, ArrayList, Vector

public abstract class **AbstractList**
extends AbstractCollection
implements List

This class provides a skeletal implementation of the `List` interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), `AbstractSequentialList` should be used in preference to this class.

Question 9. (4 points) Suppose we are sorting a list of integers using **selection sort** and part way through the sort, the list has the following contents:

0			k					
1	4	9	42	17	12	55	142	90

Fill in the diagrams below to show the contents of the list after the next two steps of the selection sort, where a step means expanding the size of the sorted part of the list by 1.

1	**4**	**9**	**12**	**17**	**42**	**55**	**142**	**90**

1	**4**	**9**	**12**	**17**	**42**	**55**	**142**	**90**

Question 10. (1 point) The Java class Math contains the following member definition:

```
public static final double PI = 3.14159265358979323846;
```

What is the significance of the word "`final`" in the above definition? (Circle the letter(s) of the correct answer(s))

[✓ a.] The value of `PI` cannot be changed after it has been initialized.

b. `PI` is a class variable, not an instance variable

c. The use of "`final`" means that this question is part of the final exam. If it were on an earlier exam, we would use "`midterm`" instead.

d. The variable `PI` is visible to client code outside the class.

Question 11. (3 points) Suppose we have a sorted list that contains the following 12 words (i.e., the size of the list is 12):

0	1	2	3	4	5	6	7	8	9	10	11
ant	bee	beetle	bug	cricket	critter	grass	rabbit	rat	snake	spider	weed

Write down the words that would be examined in the order they are examined during a binary search of the list for the word "cricket". We suggest that you hand trace the algorithm (values of L, R, and mid) both to get the question right and so we can award appropriate partial credit if something isn't quite right. Recall that in the version of binary search presented in class, initially L = -1, R = the size of the list (12 in this case), and the search does not terminate until L+1 == R.

critter beetle bug cricket

Question 12. (10 points) The weather service has hired you to do a small programming job for them. They have data about rainfall over a period of days measured at several locations. What they would like you to do is to write a method to compute the average rainfall at each location.

More specifically, the data is stored in a 2-D array. Each row represents the amount of rainfall in inches on a particular day. There is one column for each location. For example, the following array shows rainfall at 4 different locations measured over 6 days.

0.0	0.1	0.1	0.3
0.0	0.2	1.0	0.5
0.3	0.1	0.8	0.4
0.0	0.3	0.6	0.2
0.1	0.0	0.0	0.0
0.2	0.0	0.5	0.2

In this example, the first location had no rainfall for the first two days, 0.3 inches on the next day, 0.0 the next, and 0.1 and 0.2 inches on the last two days.

Complete the definition of method `averageRainfall` on the next page so it returns a one-dimensional array where each element in the result contains the average of the rainfall at the location whose data is stored in the corresponding column of the original array. In the above example, `averageRainfall` should return an array with 4 elements, where the first element (element [0]) is the average of the entries in the first column, etc.

You should assume that the original array is rectangular, i.e., each row has the same number of elements. Recall that if `a` is a 2-D array, `a.length` is the number of rows in the array and `a[k].length` is the number of columns in row `k`.

Question 12. (cont.)

```
/** Return an array whose elements are the averages of the
 *  individual columns in the rainfall data
 * @param rain  2-D array where each column represents rainfall
 *              information at a particular location.
 * @return a new 1-D array with the averages of each column in rain
 */
public double[] averageRainfall(double[][] rain) {

     double[] avg = new double[rain[0].length];

     // In java, array elements are initialized to 0.0 when
     // the array is allocated, so we don’t need to explicitly
     // initialize them here.  But no harm is done, and maybe it’s a
     // bit clearer if the elements of avg were explicitly set to 0.0

     for (int r = 0; r < rain.length; r ++) {
        for (int c = 0; c < rain[r].length; c++) { // or c<rain[0].length
           avg[c] = avg[c] + rain[r][c];
        }
     }
     for (int k = 0; k < avg.length; k++) {   // or k < rain[0].length
        avg[k] = avg[k] / rain.length;
     }
     return avg;
}
```

Another even simpler solution would be to process the columns one at a time. If we do that, we can replace the above loops by the following.

```
     for (int c = 0; c < rain[0].length; c++) {
        for (int r = 0; r < rain.length; r++) {
           avg[c] = avg[c] + rain[r][c];
        }
        avg[c] = avg[c] / rain.length;
     }
```