*8.26 (*Row sorting*) Implement the following method to sort the rows in a two-dimensional array. A new array is returned and the original array is intact.

```
public static double[][] sortRows(double[][] m)
```

Write a test program that prompts the user to enter a 3×3 matrix of double values and displays a new row-sorted matrix. Here is a sample run:

```
Enter a 3-by-3 matrix row by row:

0.15 0.875 0.375 Penter

0.55 0.005 0.225 Penter

0.30 0.12 0.4 Penter

The row-sorted array is

0.15 0.375 0.875

0.005 0.225 0.55

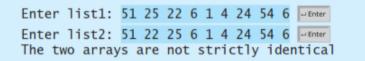
0.12 0.30 0.4
```

8.28 (Strictly identical arrays) The two-dimensional arrays m1 and m2 are strictly identical if their corresponding elements are equal. Write a method that returns true if m1 and m2 are strictly identical, using the following header:

```
public static boolean equals(int[][] m1, int[][] m2)
```

Write a test program that prompts the user to enter two 3×3 arrays of integers and displays whether the two are strictly identical. Here are the sample runs.

```
Enter list1: 51 22 25 6 1 4 24 54 6 Finter
Enter list2: 51 22 25 6 1 4 24 54 6 Finter
The two arrays are strictly identical
```



***9.6** (*Stopwatch*) Design a class named **StopWatch**. The class contains:

- Private data fields **startTime** and **endTime** with getter methods.
- A no-arg constructor that initializes **startTime** with the current time.
- A method named **start()** that resets the **startTime** to the current time.
- A method named stop() that sets the endTime to the current time.
- A method named **getElapsedTime()** that returns the elapsed time for the stopwatch in milliseconds.

Draw the UML diagram for the class and then implement the class. Write a test program that measures the execution time of sorting 100,000 numbers using selection sort.

9.7 (*The Account class*) Design a class named Account that contains:

- A private int data field named id for the account (default 0).
- A private double data field named balance for the account (default 0).
- A private double data field named annualInterestRate that stores the current interest rate (default 0). Assume all accounts have the same interest rate.
- A private Date data field named dateCreated that stores the date when the account was created.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id and initial balance.
- The accessor and mutator methods for id, balance, and annualInterestRate.
- The accessor method for dateCreated.
- A method named **getMonthlyInterestRate()** that returns the monthly interest rate.
- A method named **getMonthlyInterest()** that returns the monthly interest.
- A method named withdraw that withdraws a specified amount from the account.
- A method named **deposit** that deposits a specified amount to the account.

Draw the UML diagram for the class and then implement the class. (*Hint*: The method **getMonthlyInterest(**) is to return monthly interest, not the interest rate. Monthly interest is **balance** * **monthlyInterestRate**. **monthlyInterestRate** is a percentage, e.g., like 4.5%. You need to divide it by 100.)

Write a test program that creates an **Account** object with an account ID of 1122, a balance of \$20,000, and an annual interest rate of 4.5%. Use the **withdraw** method to withdraw \$2,500, use the **deposit** method to deposit \$3,000, and print the balance, the monthly interest, and the date when this account was created.

**9.13 (The Location class) Design a class named Location for locating a maximal value and its location in a two-dimensional array. The class contains public data fields row, column, and maxValue that store the maximal value and its indices in a two-dimensional array with row and column as int types and maxValue as a double type.

Write the following method that returns the location of the largest element in a two-dimensional array:

```
public static Location locateLargest(double[][] a)
```

The return value is an instance of **Location**. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array. Here is a sample run:



```
Enter the number of rows and columns in the array:

23.5 35 2 10 —Enter

4.5 3 45 3.5 —Enter

35 44 5.5 9.6 —Enter

The location of the largest element is 45 at (1, 2)
```