import java.util.*;

class Board {
    int ar[][];

    // creates a nxn size board
    public Board(int n) {
        ar = new int[n][n];

        // fill in array with random ints in the range [0, 100]
        Random generator = new Random();
        for (int i=0; i < n; i++)
            for (int j=0; j < n; j++)
                ar[i][j] = generator.nextInt(101);
    }

    // display the contents of the board
    public void print() {
        for (int[] r : ar) { // for all rows
            for (int c : r) // for every element in current row
               System.out.print(c + " ");
            System.out.println();
        }
    }

    // find max element in given row
    int findMaxInRow(int row) {
        int max = Integer.MIN_VALUE;
        for (int i=0; i < ar[row].length; i++)
            if (max < ar[row][i])
                max = ar[row][i];
        return max;
    }

    // find max element in given column
    }
int findMaxInColumn(int col) {
    int max = Integer.MIN_VALUE;

    // array is square nxn so same length in both dimensions
    int n = ar[0].length;
    for (int row=0; row < n; row++)
        if (max < ar[row][col])
            max = ar[row][col];

    return max;
}

/* return max across all diagonals, i.e. return the max
of all array elements */
int findMaxInDiagonal() {
    int max = Integer.MIN_VALUE;

    for (int[] r : ar) // for all rows
        for (int c : r) // for every element in current row
            if (max < c)
                max = c;

    return max;
}

public class TwoDimensionalTest {
    public static void main(String[] args) {
        Board board1 = new Board(5);
        board1.print();

        // find max value in third row (first row is at 0)
        System.out.println("\nmax in row 2: " + board1.findMaxInRow(2));

        // find max value in third column (first column is at 0)
        System.out.println("\nmax in column 2: " + board1.findMaxInColumn(2));

        // find max in all diagonals (max of array)
        System.out.println("\nmax in board: " + board1.findMaxInDiagonal());
    }
}

Note that when we traverse all diagonals of the board we traverse all elements of the array. Therefore, finding the maximum value found among all diagonals is equivalent to find the maximum element in the whole board.
Manipulating Arrays

The program displays:

a contains: -2 -1 0 1 2 3 4 5 6 7
b contains: -2 -1 0 1 2 3 4 5 6 7
c contains:
-2 -1 0 1 2 3 4 5 6 7
2 3 4 5 6

Generics - ArrayLists

import java.util.ArrayList;

class Book {
    public String author;
    public String title;
}

class Library {
    ArrayList<Book> books = new ArrayList<Book>();

    // populate arraylist books with 4 book objects
    void populate() {
        for (int i=1; i <= 4; i++) {
            // create a new book with "some" data
            Book b = new Book();
            b.author = "author" + i;
            b.title = "title" + i;

            // add book in list
            books.add(b);
        }
    }

    // print details of all books in the library
    void displayAllBooks() {
        for (int i=0; i < books.size(); i++) {
            Book b = books.get(i);
            System.out.println("Book: author=" + b.author + ", title=" + b.title);
        }
    }
}

public static void main(String[] args) {
    Library lib = new Library();
    lib.populate();
lib.displayAllBooks();
}
}

Non-parameterised ArrayLists

1. • d = list.get(0);: The get method of ArrayList returns an Object type which cannot be assigned to a double (or Double) without a cast. 
   Correct it as:
   
   d = (Double) list.get(0);

   The returned Double object will be auto-unboxed to a double.

   • Double d2 = (Integer) list.get(1);: An Integer object cannot be assigned to a Double reference variable.
   Modify the left hand side of the assignment to be of Integer type:
   
   Integer d2 = (Integer) list.get(1);

   • Book b2 = list.get(2);: The get method of ArrayList returns an Object type which cannot be assigned to Book without a cast.
   Correct is as:
   
   Book b2 = (Book) list.get(2);

   • Book b3 = (Book) list.get(1);: Although this line compiles, during execution an exception will be thrown.
   This is because at index position 1 of the arraylist, an Integer object is held. During runtime, an attempt to cast it and assign it to a Book type will generate an exception.

2. The corrected program is:

   import java.util.ArrayList;
   
   class Book {
       public String author;
       public String title;
   }
   
   public class ContainerTest {
       public static void main(String[] args) {
           ArrayList list = new ArrayList();

           list.add(3.4);
           list.add(new Integer(4));

           Book myBook = new Book();
           list.add(myBook);

           double d;
           d = (Double) list.get(0);
3. See comments above.

Static Methods

class CalcManager {
    public static boolean isEven(int n) {
        if (n % 2 == 0)
            return true;
        else
            return false;
    }

    public static int cube(int n) {
        return n*n*n;
    }

    public static double add(double...x) {
        double sum = 0.0;
        for (double e : x)
            sum = sum + e;
        return sum;
    }
}

class CalcManagerTest {
    public static void main(String[] args) {
        double sum1 = CalcManager.add(2.1, 5.6);
        System.out.println("sum1 = " + sum1);

        double sum2 = CalcManager.add(7.9, 1.3);
        System.out.println("sum2 = " + sum2);

        System.out.println("isEven(4): " + CalcManager.isEven(4));
        System.out.println("isEven(7): " + CalcManager.isEven(7));

        System.out.println("cube(3): " + CalcManager.cube(3));
        System.out.println("cube(10): " + CalcManager.cube(10));
    }
}
The program displays:

\[
\begin{align*}
\text{sum1} &= 7.699999999999999 \\
\text{sum2} &= 9.200000000000001 \\
\text{isEven}(4): \text{true} \\
\text{isEven}(7): \text{false} \\
\text{cube}(3): 27 \\
\text{cube}(10): 1000
\end{align*}
\]

### Static Fields

1. The output of the program is:
   
   e1 contains: Name: John, 77 Exhibition Road London  
   e2 contains: Name: George, 77 Exhibition Road London  
   e3 contains: Name: Helen, 77 Exhibition Road London

   After e1.company_address = a2  
   e1 contains: Name: John, 5 Guilford Road York  
   e2 contains: Name: George, 5 Guilford Road York  
   e3 contains: Name: Helen, 5 Guilford Road York

   Memory location for e1.company_address object: Address@15ff48b  
   Memory location for e2.company_address object: Address@15ff48b  
   Memory location for e3.company_address object: Address@15ff48b

   e1.numberOfEmployees: 3  
   e2.numberOfEmployees: 3  
   e3.numberOfEmployees: 3

3. Since `company_address` is static in `Employee` it is shared among all objects of the class. Therefore it can only have one value for all objects, and employees working for different companies cannot be created in this program.