

CS250 Winter 2003 Lecture 5: Exception handling, Static methods

```
public static <return type> <method name>
    (<paratype1> <paraname1>
     <paratype2> <paraname2>, ...)
}
```

Breakdown:

`public`: declares which classes are allowed to use this method. (could also be '`private`' or '`protected`')

`static`: declares this method as static (i.e does not operate on an object)

`<return type>`: specifies the return type. Could be any object type or primitive type, or an array

Jan 13, 03 22:19

except.java

Page 1/1

```
import java.io.*;
import java.lang.*;

/* Example of exception handling. Notice that I don't have
   to throw the NumberFormatException, since it handled inside
   main. */
public class except {

    public static void main(String[] args)
        throws IOException
    {

        BufferedReader stndin;
        stndin = new BufferedReader(new InputStreamReader(System.in));

        System.out.print( "Enter an integer: " );
        System.out.flush();

        String line = stndin.readLine();

        if (line == null) {
            System.out.println( "No input!" );
            return;
        }

        // Parse menu choice

        int choice;
        while(true) {
            try {
                choice = Integer.parseInt(line);
                break;
            }
            catch (NumberFormatException e) {
                System.out.println("Bad input.try again!");
                line = stndin.readLine();
            }
        }

        System.out.println( "Phew. It's about time!" );

    }
}
```

Jan 16, 03 17:27

methodEx.java

Page 1/1

```
/* This class contains several simple static methods, meant for
demonstration. Example method calls are provided in main() */

public class methodEx {

    // returns the square x
    public static int square(int x) {
        return x*x;
    }

    /* returns the larger of two values. */
    public static int max(int x, int y) {
        if (x > y)
            return x;
        else
            return y;
    }

    public static int max(int x, int y, int z) {
        if (x > y && x > z)
            return x;
        else if (y > x && y > z)
            return y;
        else
            return z;
    }

    /* returns the average of two numbers */
    public static float average(float f, float g) {
        return (f+g)/2;
    }

    /* a method with no inputs and no return value. */
    public static void printMenu() {
        System.out.println("1. Do this");
        System.out.println("2. Do that");
        System.out.println("3. Quit");
    }

    public static void main(String[] args) {
        // Several calls to the square function
        System.out.println("The square of 20 is: " +square(20));
        int x = 2;
        System.out.println("The square of x is: " +square(x));
        int y = square(5);
        System.out.println("5^2 is:"+ y);
        System.out.println("5^4 is:"+ square(square(5)));
        System.out.println();

        System.out.println("max(x,y) is: "+ max(x,y));

        /* here we call a function that has no return value */
        printMenu();

        float f = (float) 3.14, g = (float)1.1;
        System.out.println("The average of 3.14 and 1.1 is:"+average(f,g));
    }
}
```



Jan 13, 03 22:46

arrayOutputEx.java

Page 1/1

```

/* Some useful routines that produce arrays as output */
public class arrayOutputEx {

/* In this example a[] is used as both input and output */
public static void sort(int a[]) {
    if (a.length < 1) return;

    for(int i = 0; i < a.length; i++) {
        int min = a[i];
        int minidx = i;

        for(int j = i+1; j < a.length; j++) {
            if (a[j] < min) {
                min = a[j];
                minidx = j;
            }
        }

        int temp = a[minidx];
        a[minidx] = a[i];
        a[i] = temp;
    }
}

/* Returns array containing the first n even numbers. */
public static int[] evenNumbers(int n) {
    int a[] = new int[n];

    for (int i = 0; i < n; i++) {
        a[i] = 2*i;
    }
    return a;
}

public static void main(String[] args) {
    int nums[] = {3,4,6,1,2,6,7,8};

    System.out.println("original sum of nums");
    for (int i=0; i < nums.length; i++) {
        System.out.println(nums[i]);
    }

    sort(nums);

    System.out.println("After sorting:");
    for (int i=0; i < nums.length; i++) {
        System.out.println(nums[i]);
    }

    System.out.println("Even numbers:");
    int even[] = evenNumbers(10);
    for (int i=0; i < nums.length; i++) {
        System.out.println(even[i]);
    }
}
}

```

Dec 19, 02 10:27

arrayInputEx.java

Page 1/1

```
/* Some useful function that take arrays as input */
public class arrayInputEx {

    // return the index of the first occurrence of s in a[]
    public static int index(String a[], String s) {
        for(int i=0; i < a.length; i++) {
            if (a[i].equals(s))  return i;
        }
        return -1;
    }

    // compute the sum of the entries in a[]
    public static int sum(int a[]) {
        int count =0;
        for(int i=0; i < a.length; i++) {
            count+=a[i];
        }
        return count;
    }

    public static void main(String args[]) {
        int numbers[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        System.out.println("The sum of the elements in numbers[] is:"
                           + sum(numbers));

        String names[] = {"Fred", "Sue", "Joe", "Bob"};
        System.out.println("Sue's name is at index: "+index(names, "Sue"));

    }
}
```

Jan 14, 03 9:59

Merge.java

Page 1/2

```

public class Merge{
    /* As part of merge sort, merge two semi sorted lists */
    public static void mergeList(int a[], int l_start,
                                 int l_end, int r_end) {

        int temp[] = new int[r_end+1];
        int t_idx = 0, l_idx = l_start, r_idx = l_end+1;

        // merge two arrays into temp array
        while (l_idx <= l_end || r_idx <= r_end) {
            /* first two conditions correspond to the case
             where one list is already empty. */
            if (l_idx > l_end) {
                temp[t_idx] = a[r_idx++];
            }
            else if (r_idx > r_end) {
                temp[t_idx] = a[l_idx++];
            }
            else {
                if (a[l_idx] < a[r_idx])
                    temp[t_idx] = a[l_idx++];
                else
                    temp[t_idx] = a[r_idx++];
            }
            t_idx++;
        }

        /* list merged. copy back to original place. */
        for (int i= 0; i < t_idx; i++) {
            a[l_start + i] = temp[i];
        }
    }

    //recursively merge the items between a[min] and a[max]
    public static void rmergeSort(int a[], int min, int max) {

        if (max - min == 0) {
            /* the one-element list is trivially sorted */
        }
        else if (max - min == 1) {
            /* Another easy case. Swap if out of order */
            if (a[min] > a[max]) {
                int temp = a[max];
                a[max] = a[min];
                a[min] = temp;
            }
        }
        else {
            /* Ok, here is the general case */
            int mid = (min + max) / 2;

            rmergeSort(a, min, mid);
            rmergeSort(a, mid+1, max);
            mergeList(a, min, mid, max);
        }
    }

    public static void mergeSort(int a[]) {
        rmergeSort(a, 0, a.length-1);
    }

    /* I test the mergesort routine here */
    public static void main(String args[]) {
        int nums[] = {1,4, 5,2, 6, 3, 2, 7, 7, 8,3,32, 25};

        mergeSort(nums);
    }
}

```

00

Jan 14, 03 9:59

Merge.java

Page 2/2

```

        for (int i=0; i < nums.length; i++) {
            System.out.println(nums[i]);
        }
    }
}

```