Outline

Method OverLoading

printf method

Arrays

- Declaring and Using Arrays
- Arrays of Objects
- ➡ Array as Parameters
- Variable Length Parameter Lists

`split()` Method from String Class

`Integer` & `Double` Wrapper Classes

Two-Dimensional Arrays

Arrays as Parameters

- An entire array can be passed as a parameter to a method
- Like any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other
- Therefore, changing an array element within the method changes the original
- An individual array element can be passed to a method as well, in which case the type of the formal parameter is the same as the element type

Examples of Array as Parameters

[Array4.java]

[MakeHot.java]

[ArrParametersAndReturns.java]

Outline

- **Method OverLoading**
- **printf method**
- **Arrays**
  - **Declaring and Using Arrays**
  - **Arrays of Objects**
  - **Array as Parameters**
  - ➡ **Variable Length Parameter Lists**
- **`split()` Method from String Class**
- **`Integer` & `Double` Wrapper Classes**
- **Two-Dimensional Arrays**

Variable Length Parameter Lists

- Suppose we wanted to create a method that processed a different amount of data from one invocation to the next
- For example, let's define a method called `average` that returns the average of a set of integer parameters

```
// one call to average three values
mean1 = average (42, 69, 37);

// another call to average seven values
mean2 = average (35, 43, 93, 23, 40, 21, 75);
```

Variable Length Parameter Lists

- Using special syntax in the formal parameter list, we can define a method to accept any number of parameters of the same type
- For each call, the parameters are automatically put into an array for easy processing in the method

Indicates a variable length parameter list

```
public double average (int ... list)
{
   // whatever
}
```

element type

array name

Variable Length Parameter Lists

```
public double average (int ... list)
{
   double result = 0.0;

   if (list.length != 0)
   {
      int sum = 0;
      for (int num : list){
         sum += num;
      }
      result = (double)sum / list.length;
   }

   return result;
}
```

Example

- See `VarArgsDemo.java`

Example

Write method called `distance` that accepts a variable number of doubles (which each represent the distance of one leg of a trip) and returns the total distance of the trip.

See **`Trip.java`**

Variable Length Parameter Lists

- A method that accepts a variable number of parameters can also accept other parameters, but variable number of parameters can exist only once and as the last parameter
- The following method accepts an `int`, a `String` object, and a variable number of `double` values into an array called `nums`

```
public void test (int count, String name,
                  double ... nums)
{
   // whatever
}
```

Outline

`split()` Method from `String` Class

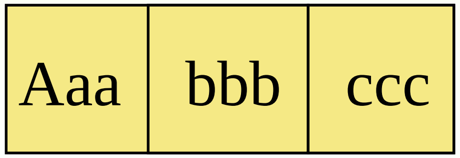
`public String[] split(String delimiter)`

- Method takes a String delimiter
- Returns a String array
- The **split** method in the **String** class returns an array of strings consisting of the substrings split by the delimiters.

```
String str = "Aaa, bbb, ccc";
String[] arr =  str.split(", ");
```

delimiter

`arr =`



Examples:

```
String str = "Java#HTML#Perl";
String [] tokens = str.split("#");

for (String token: tokens)
      System.out.println (token.toUpperCase());
```

Examples:

```
String str = “Java#HTML#Perl”;
String [] tokens = str.split("#");

for (String token: tokens)
      System.out.println (token.toUpperCase());
```

Generates the following output:

```
JAVA
HTML
PERL
```

Examples:

```
String str = "Hi  How are you?";
String [] tokens = str.split("\t");

for (int i=0; i < tokens.length; i++)
      System.out.println (tokens[i]);
```

Examples:

```
String str = "Hi  How are you?";
String [] tokens = str.split("\t");

for (int i=0; i < tokens.length; i++)
      System.out.println (tokens[i]);
```

Generates the following output:

```
Hi
How are you?
```

Outline

- Method OverLoading
- printf method
- Arrays
  - Declaring and Using Arrays
  - Arrays of Objects
  - Array as Parameters
  - Variable Length Parameter Lists
- `split()` Method from String Class
- ➡ `Integer` & `Double` Wrapper Classes
- Two-Dimensional Arrays

Wrapper Classes

- A wrapper class represents a particular primitive type as an object. For example, Integer class represents a simple integer value.

- An Integer object may be created as

```
Integer obj = new Integer(40);
```

- All wrapper classes are in the java.lang package (no need to import).

Wrapper Classes in the `java.lang` Package

Primitive type	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean
void	Void

`Integer` Class

Useful methods from the Integer class:

- **`Integer (int value)`** `: creates an Integer object storing the specified value`
- **`Integer (String str)`** `: creates an Integer object storing the integer value extracted from the string, str`
- **`static int parseInt(String str)`** `: returns the int corresponding to the value stored in the specified string, str`
- **`static String toBinaryString(int num)`** `: returns a string representation of the specified integer value in base 2`
- **`static String toString(int num)`** `: returns a string representation of the specified integer value in base 10`
- **`static Integer valueOf(String str)`** `: returns a Integer object holding the int value represented by the argument string str.`

Constants from the Integer class:

Integer.MIN_VALUE returns the minimum int value, -2^{31}

Integer.MAX_VALUE returns the maximum int value, $2^{31} -1$

`Double` Class

Some methods from the Double class:

- **`Double (double value)`** `: creates a Double object storing the specified value`
- **`Double (String str)`** `: creates a Double object storing the double value in string, str`
- **`static double parseDouble(String str)`** `: returns the double corresponding to the value stored in the specified string, str`
- **`static String toString(double num)`** `: returns a string representation of the specified double value`
- **`static Double valueOf(String str)`** `: returns a Double object holding the double value represented by the argument string str.`

Constants from the Double class:

`Double.MIN_VALUE` returns the minimum double value

`Double.MAX_VALUE` returns the maximum double value

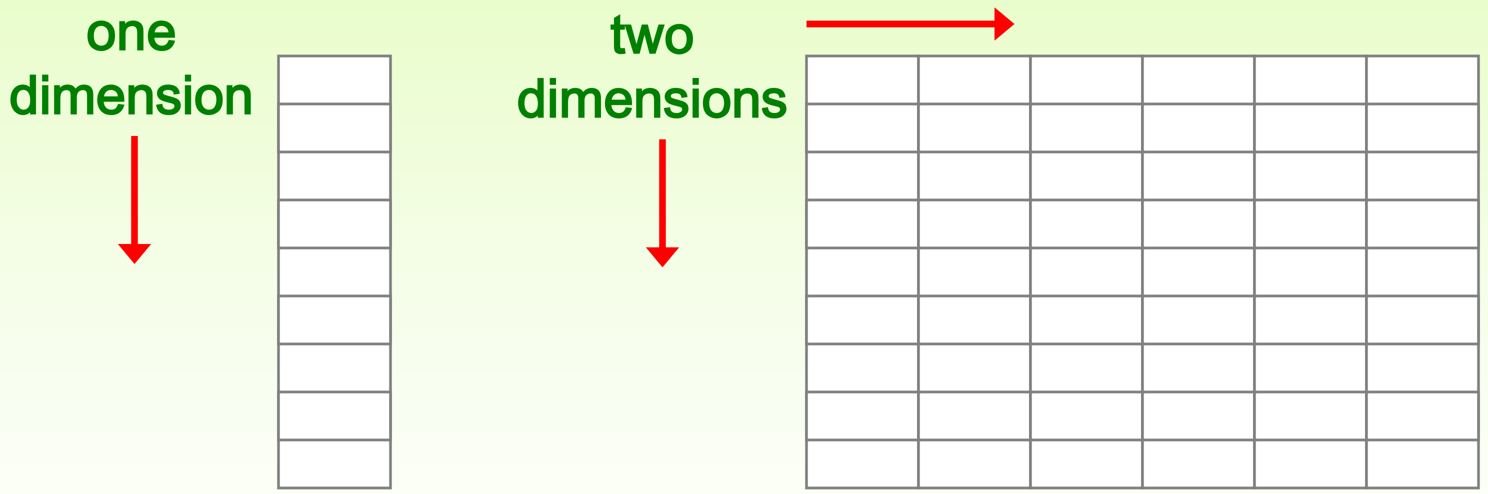
Example for split() and reading from a text file

- See `WrapperSplitFile.java`

Outline

Two-Dimensional Arrays

- A *one-dimensional array* stores a list of elements
- A *two-dimensional array* can be thought of as a table of elements, with rows and columns



Two-Dimensional Arrays

- To be precise, in Java a two-dimensional array is an array of arrays
- A two-dimensional array is declared by specifying the size of each dimension separately:

```
int[][] table = new int[12][50];
```

- A array element is referenced using two index values:

```
value = table[3][6]
```

- The array stored in one row can be specified using one index

Two-Dimensional Arrays

Expression	Type	Description
`table`	**`int[][]`**	2D array of integers, or array of integer arrays
`table[5]`	**`int[]`**	array of integers
`table[5][12]`	**`int`**	integer

Two-Dimensional Arrays

```
int[][]  matrix = new int[2][7]; //creates a 2-by-7 array
```

0	0	0	0	0	0	0
0	0	0	0	0	0	0

```
matrix[0][2] = 5;
```

0	0	5	0	0	0	0
0	0	0	0	0	0	0

Multidimensional Arrays

- You can think of it as array of arrays

```
for (int i=0; i < 2; i++){
    for (int j=0; j<7; j++){
        matrix[i][j]=1;
    }
}
```

1	1	1	1	1	1	1
1	1	1	1	1	1	1

Two-Dimensional Arrays

- See `TwoDArray.java`

```java
//********************************************************************
//  TwoDArray.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a two-dimensional array.
//********************************************************************

public class TwoDArray
{
   //-----------------------------------------------------------------
   //  Creates a 2D array of integers, fills it with increasing
   //  integer values, then prints them out.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int[][] table = new int[5][10];

      // Load the table with values
      for (int row=0; row < table.length; row++)
         for (int col=0; col < table[row].length; col++)
            table[row][col] = row * 10 + col;

      // Print the table
      for (int row=0; row < table.length; row++)
      {
         for (int col=0; col < table[row].length; col++)
            System.out.print (table[row][col] + "\t");
         System.out.println();
      }
   }
}
```

```
//********************************************************************
//  TwoDArray.java       Author: Lewis/Loftus
//
```

Output

```
0       1       2       3       4       5       6       7       8       9
10      11      12      13      14      15      16      17      18
        19
20      21      22      23      24      25      26      27      28
        29
30      31      32      33      34      35      36      37      38
        39
40      41      42      43      44      45      46      47      48
        49
```

```
   public static void main (String[] args)
   {
      int[][] table = new int[5][10];

      // Load the table with values
      for (int row=0; row < table.length; row++)
         for (int col=0; col < table[row].length; col++)
            table[row][col] = row * 10 + col;

      // Print the table
      for (int row=0; row < table.length; row++)
      {
         for (int col=0; col < table[row].length; col++)
            System.out.print (table[row][col] + "\t");
         System.out.println();
      }
   }
}
```

Example & Exercise

- See `GradeExam.java`
- **Exercise:** Modify the code such that there are two methods:
  - **`gradeAStudent : grades one student`**
  - **`gradeAllStudents: grades all students`**

Example

- See `FindNearestPoints.java`

Two-Dimensional Arrays

- See `SodaSurvey.java`

```java
//********************************************************************
//  SodaSurvey.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a two-dimensional array.
//********************************************************************

import java.text.DecimalFormat;

public class SodaSurvey
{
   //-----------------------------------------------------------------
   //  Determines and prints the average of each row (soda) and each
   //  column (respondent) of the survey scores.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int[][] scores = { {3, 4, 5, 2, 1, 4, 3, 2, 4, 4},
                         {2, 4, 3, 4, 3, 3, 2, 1, 2, 2},
                         {3, 5, 4, 5, 5, 3, 2, 5, 5, 5},
                         {1, 1, 1, 3, 1, 2, 1, 3, 2, 4} };

      final int SODAS = scores.length;
      final int PEOPLE = scores[0].length;

      int[] sodaSum = new int[SODAS];
      int[] personSum = new int[PEOPLE];
```

continue

continue

```
      for (int soda=0; soda < SODAS; soda++)
         for (int person=0; person < PEOPLE; person++)
         {
            sodaSum[soda] += scores[soda][person];
            personSum[person] += scores[soda][person];
         }

      DecimalFormat fmt = new DecimalFormat ("0.#");
      System.out.println ("Averages:\n");

      for (int soda=0; soda < SODAS; soda++)
         System.out.println ("Soda #" + (soda+1) + ": " +
                    fmt.format ((float)sodaSum[soda]/PEOPLE));

      System.out.println ();
      for (int person=0; person < PEOPLE; person++)
         System.out.println ("Person #" + (person+1) + ": " +
                    fmt.format ((float)personSum[person]/SODAS));
   }
}
```

continue

```
      for (int soda=0;
         for (int perso                              person++)
         {
            sodaSum[soc                             son];
            personSum[p                             [person];
         }

      DecimalFormat fmt                             "0.#");
      System.out.printl

      for (int soda=0;
         System.out.pri                             +1) + ": " +
                    fmt                             m[soda]/PEOPLE));

      System.out.printl
      for (int person=0                             son++)
         System.out.pri                             rson+1) + ": " +
                    fmt                             Sum[person]/SODAS));
   }
}
```

Output

```
Averages:

Soda #1: 3.2
Soda #2: 2.6
Soda #3: 4.2
Soda #4: 1.9

Person #1: 2.2
Person #2: 3.5
Person #3: 3.2
Person #4: 3.5
Person #5: 2.5
Person #6: 3
Person #7: 2
Person #8: 2.8
Person #9: 3.2
Person #10: 3.8
```