Strings and Arrays

Hendrik Speleers

Strings and Arrays

Università di Roma Tor Vergata

- Overview
  - Characters and strings
    - String manipulation
    - Formatting output
  - Arrays
    - One-dimensional
    - Two-dimensional
  - Container classes
    - List: ArrayList and LinkedList
    - Iterating over a list

Strings and Arrays

Università di Roma Tor Vergata

- Characters and strings
  - A character (**char** primitive)
    - Based on Unicode
    - Indicated by single quotation
  - A string (**String** object) is a container of characters
    - Indicated by double quotation
    - Operator + : concatenation of strings into a new string

```
char c = 'a';
String str = "UNICODE for a is " + (int) c;
if (c < 'W') {...}
```

Strings and Arrays

Università di Roma Tor Vergata

- Characters and strings
  - Special characters: using an escape sequence (with backslash)

Code	Meaning
\t	tab
\b	backspace
\n	newline
\r	carriage return
\'	single quote
\"	double quote
\\	backslash

Strings and Arrays

Università di Roma Tor Vergata

- Characters and strings
  - Some useful String methods

```
<integer> = <string> . length () ;
   <char> = <string> . charAt (<integer>) ;
<integer> = <string> . indexOf (<char>) ;
 <string> = <string> . substring (<integer>, <integer>) ;
<boolean> = <string> . equals (<object>) ;
```

  - Every object can be converted to a string (default)

```
 <string> = <object> . toString () ;
```

Strings and Arrays

Università di Roma Tor Vergata

- Java program: string manipulation
  - Task
    - Counting the number of words in a string
  - Algorithm
    - Running over characters
    - Counting each transition from blanc to non-blanc characters
  - Input

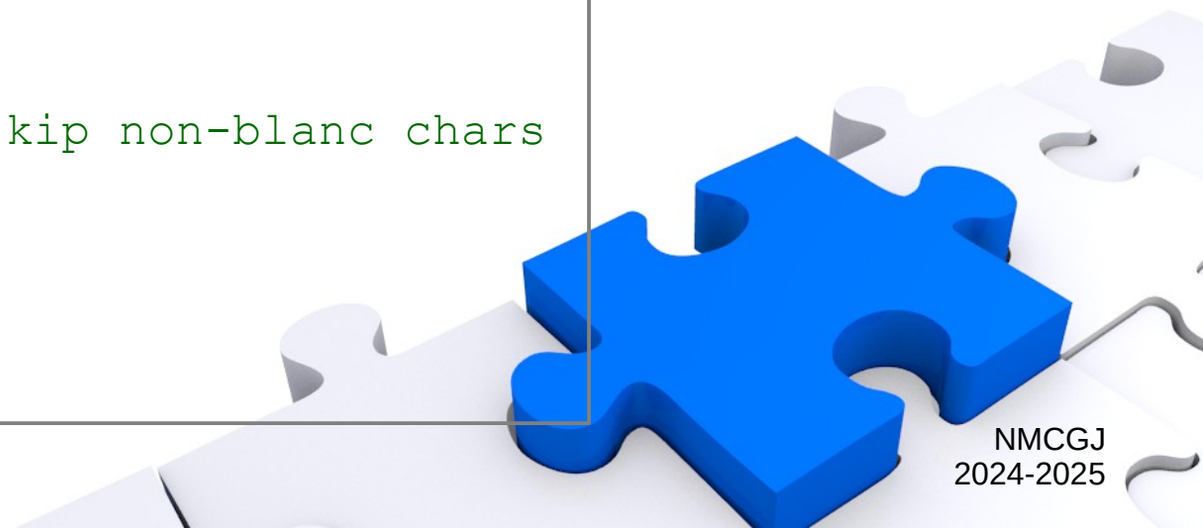
```
String text = "Twinkle twinkle little star";
```

Strings and Arrays

Università di Roma Tor Vergata

- Java program: string manipulation
  - Counting the number of words in a string

```
int i = 0, wordCount = 0;

while (i < text.length()) {

   while (text.charAt(i) == ' ') // skip blanc chars
      i++;

   while (text.charAt(i) != ' ') // skip non-blanc chars
      i++;
   wordCount++;

}
```



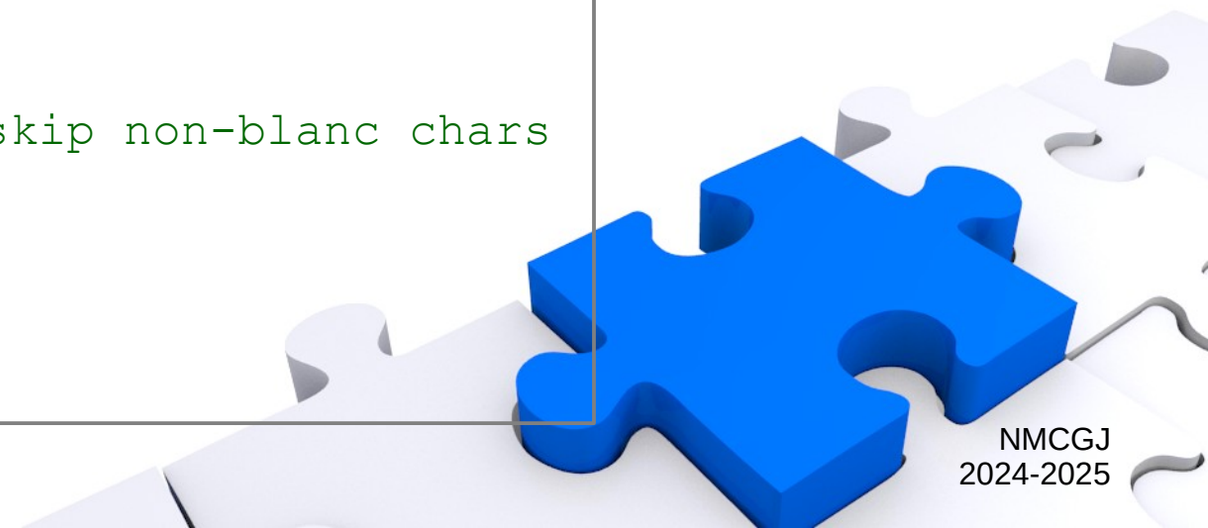
Strings and Arrays

Università di Roma Tor Vergata

- Java program: string manipulation
  - Counting the number of words in a string

```
int i = 0, wordCount = 0;

while (i < text.length()) {

   while (text.charAt(i) == ' ') // skip blanc chars
      i++;

   while (text.charAt(i) != ' ') // skip non-blanc chars
      i++;
   wordCount++;

}
```

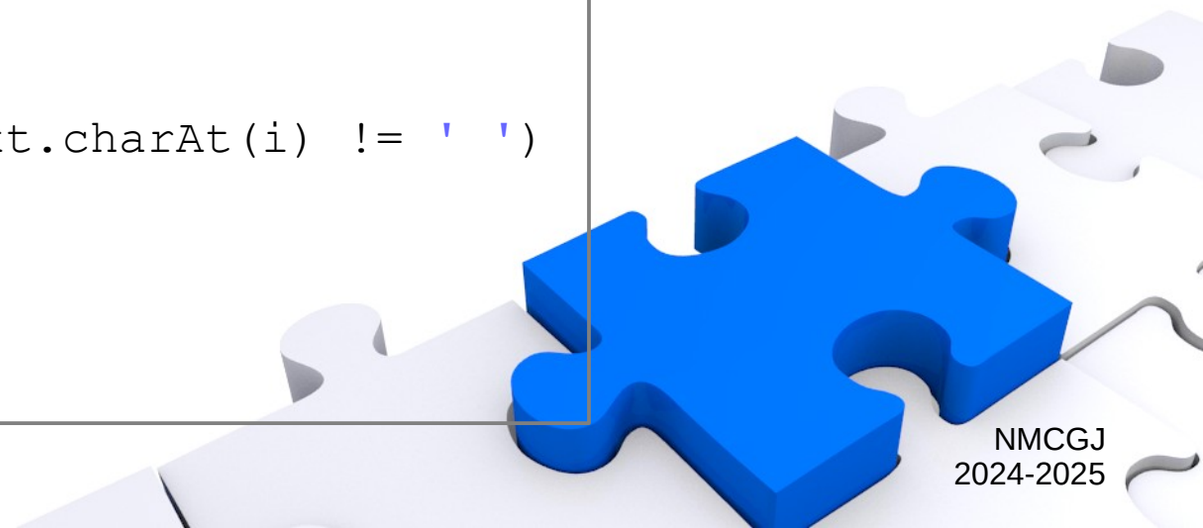
NOT CORRECT



Strings and Arrays

- Java program: string manipulation
  - Counting the number of words in a string

```
int i = 0, wordCount = 0;

while (i < text.length()) {

   while (i < text.length() && text.charAt(i) == ' ')
      i++;
   if (i < text.length()) {
      while (i < text.length() && text.charAt(i) != ' ')
         i++;
      wordCount++;
   }
}
```



Strings and Arrays

- Characters and strings
  - Formatting output (since Java SE5)
    - System.out.printf(), System.out.format(), String.format( )

```
int x = 5; double y = 5.332542;

// the old way
System.out.println("Row [" + x + " " + y + "]");
// the new way
System.out.printf("Row [%d %f]\n", x, y);
// or
System.out.format("Row [%d %f]\n", x, y);
// output
// Row [5 5.332542]
```

Strings and Arrays

- Characters and strings
  - Format specifiers

```
%<flags><width><.precision><conversion>
```

optional

- conversion

Code	Meaning
d	integer (decimal)
x	integer (hexadecimal)
f	floating point (decimal)
e	floating point (scientific)

Code	Meaning
c	character (Unicode)
s	string
b	boolean
%	literal “%”

Strings and Arrays

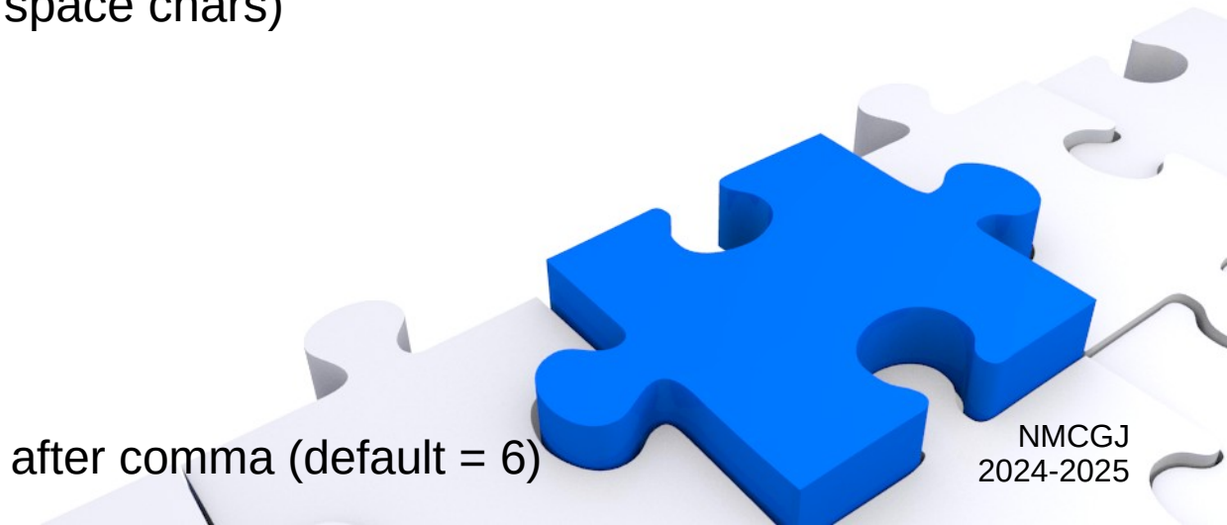
Università di Roma Tor Vergata

- Characters and strings
  - Format specifiers

```
%<flags><width><.precision><conversion>
```

optional

    - width: minimum size of a field (padding with space chars)
    - flags: controls alignment
      - default: right alignment
      - using ‘–’: left alignment
    - Precision: meaning depends on type
      - strings: maximum number of characters
      - floating points: number of decimal places after comma (default = 6)



Strings and Arrays

Università di Roma Tor Vergata

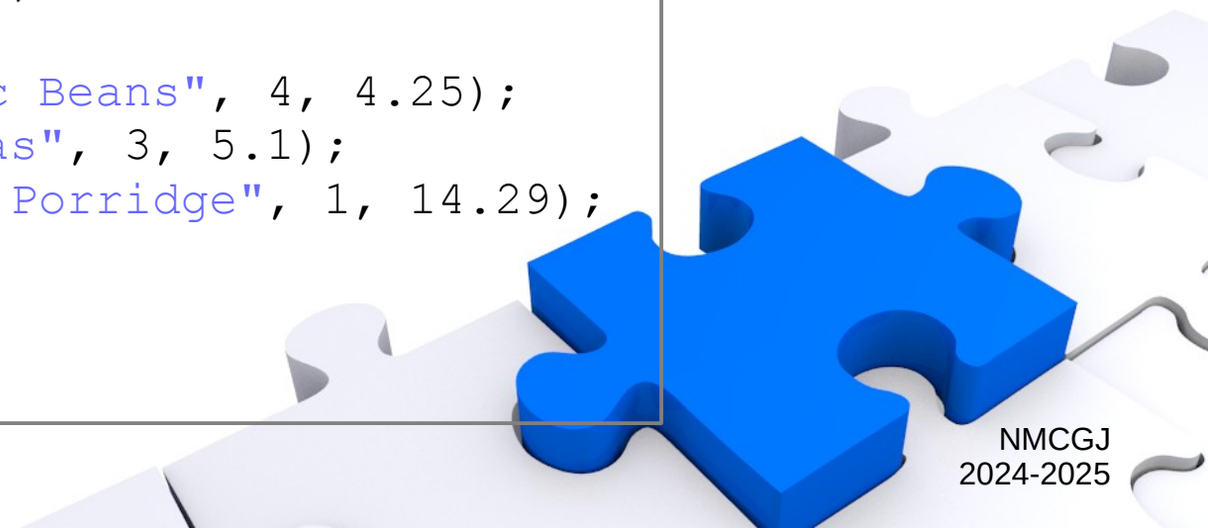
- Java program: string formatting
  - Task
    - Printing a shopping receipt to the console
  - Output

```
Item                        Qty       Price
----                        ---       -----
Jack's Magic Beans            4        4.25
Princess Peas                 3        5.10
Three Bears Porridge          1       14.29
                                      -----
Total                                 23.64
```

Strings and Arrays

- Java program: string formatting
  - Printing a shopping receipt to the console

```
public class Receipt {
    private double total = 0.0;
    ...
    public static void main(String[] args) {
        Receipt receipt = new Receipt();
        receipt.printTitle();
        receipt.printItem("Jack's Magic Beans", 4, 4.25);
        receipt.printItem("Princess Peas", 3, 5.1);
        receipt.printItem("Three Bears Porridge", 1, 14.29);
        receipt.printTotal();
    }
}
```

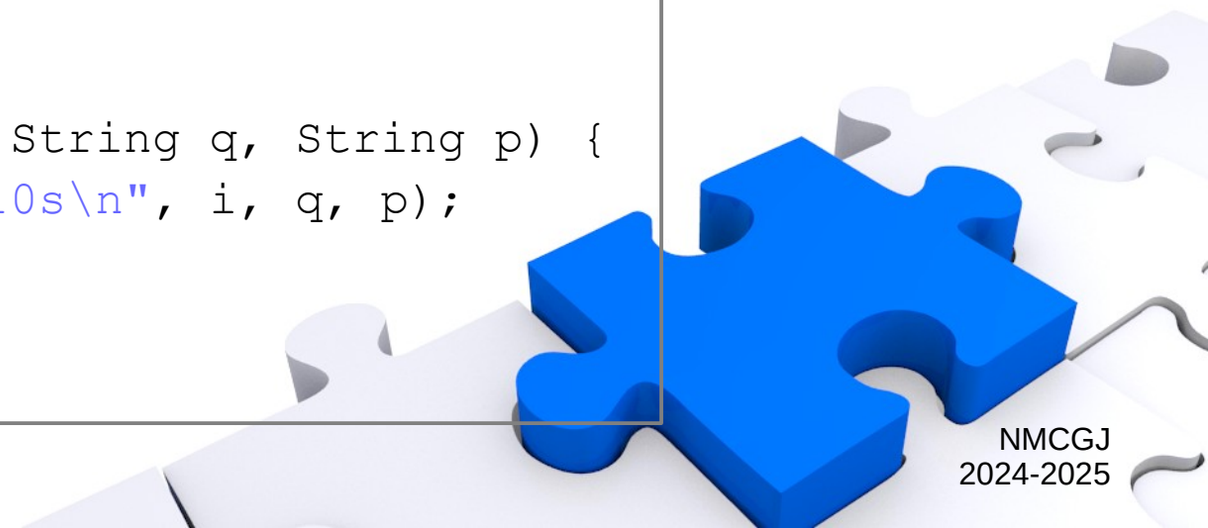


Strings and Arrays

Università di Roma Tor Vergata

- Java program: string formatting
  - Printing a shopping receipt to the console

```
public class Receipt {
    ...
    public void printTitle() {
        formatItem("Item", "Qty", "Price");
        formatItem("----", "---", "-----");
    }

    private void formatItem(String i, String q, String p) {
        System.out.format("%-20s %5s %10s\n", i, q, p);
    }
    ...
}
```

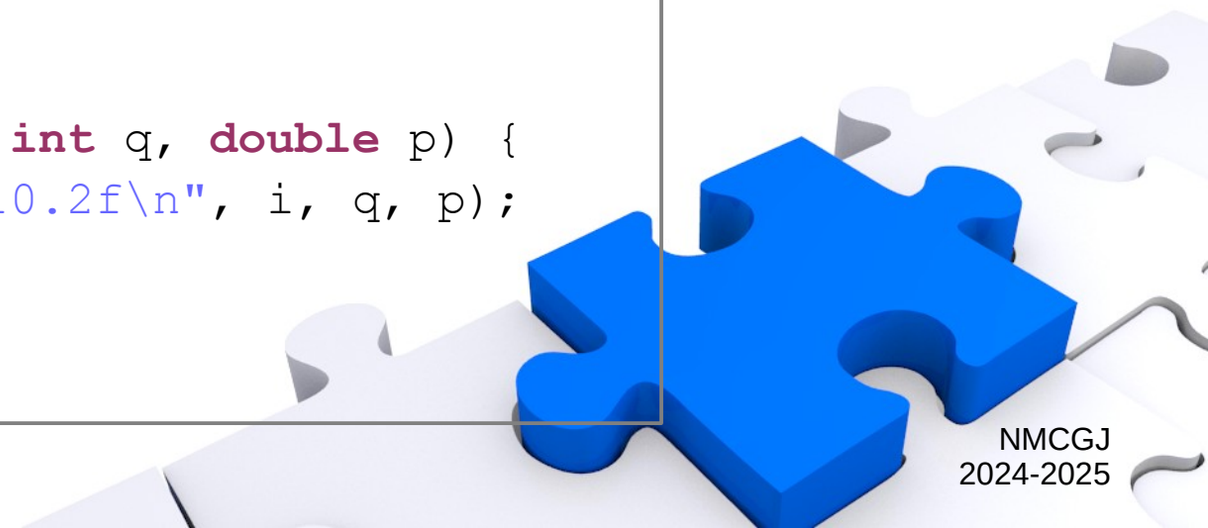


Strings and Arrays

Università di Roma Tor Vergata

- Java program: string formatting
  - Printing a shopping receipt to the console

```
public class Receipt {
    ...
    public void printItem(String i, int q, double p) {
        formatItem(i, q, p);
        total += p;
    }

    private void formatItem(String i, int q, double p) {
        System.out.format("%-20s %5d %10.2f\n", i, q, p);
    }
    ...
}
```



Strings and Arrays

Università di Roma Tor Vergata

- Java program: string formatting
  - Printing a shopping receipt to the console

```
public class Receipt {
    ...
    public void printTotal() {
        formatItem("", "", "-----");
        formatTotal("Total", total);
    }

    private void formatTotal(String s, double t) {
        System.out.format("%-20s  %15.2f\n", s, t);
    }
    ...
}
```

Strings and Arrays

Università di Roma Tor Vergata

- Arrays
  - An array is a sequence of elements of same type (primitives / objects)
  - Declaration

```
<type>[] <array name> ;
```

an array is an object: reference semantics

  - Creation

```
<array name> = new <type>[<integer>] ;
```

  - Number of elements

```
<integer> = <array name> . length ;
```

Strings and Arrays

Università di Roma Tor Vergata

- Arrays
  - Initialization

```
int nbMonths = 12;
String[] months = new String[nbMonths];
months[0]  = "January";
months[1]  = "February";
...
months[11] = "December";
```

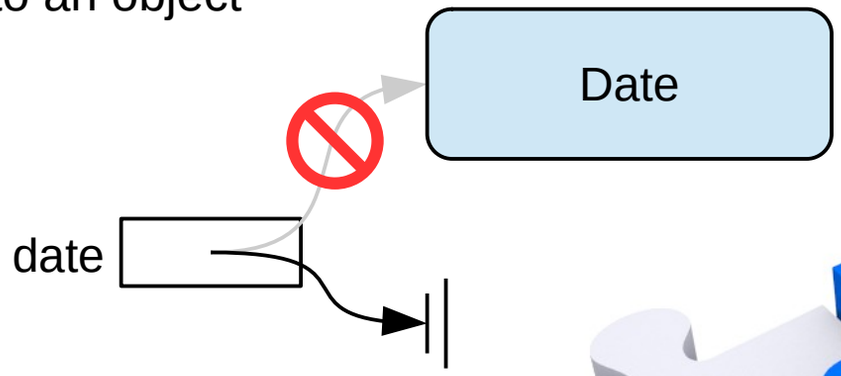
index of first position in an array is zero

```
String[] months = {"January", "February", "March",
                   "April", "May", "June", "July",
                   "August", "September", "October",
                   "November", "December"};
```

Strings and Arrays

- Arrays
  - Initialization
    - Default value for primitives: zero
    - Default value for objects: empty object
  - Creating an empty object with the **null** keyword
    - Object variable is not pointing to an object
    - Identical for any object type

```
Date date = new Date();
date = null;
```



Strings and Arrays

Università di Roma Tor Vergata

- Arrays
  - Repetition statement: foreach loop (since Java SE5)

```
for (<element> : <array name>)
   <for block>
```

if <block> consists of multiple statements, then use { }

  - Comparison between standard for and foreach

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

```
for (String m : months)
   System.out.println(m);
```

Strings and Arrays

Università di Roma Tor Vergata

- Java program: arrays
  - Task
    - Searching maximum element in an unsorted array
  - Algorithm
    - Running over elements
    - Comparing current element > maximum value so far
  - Input

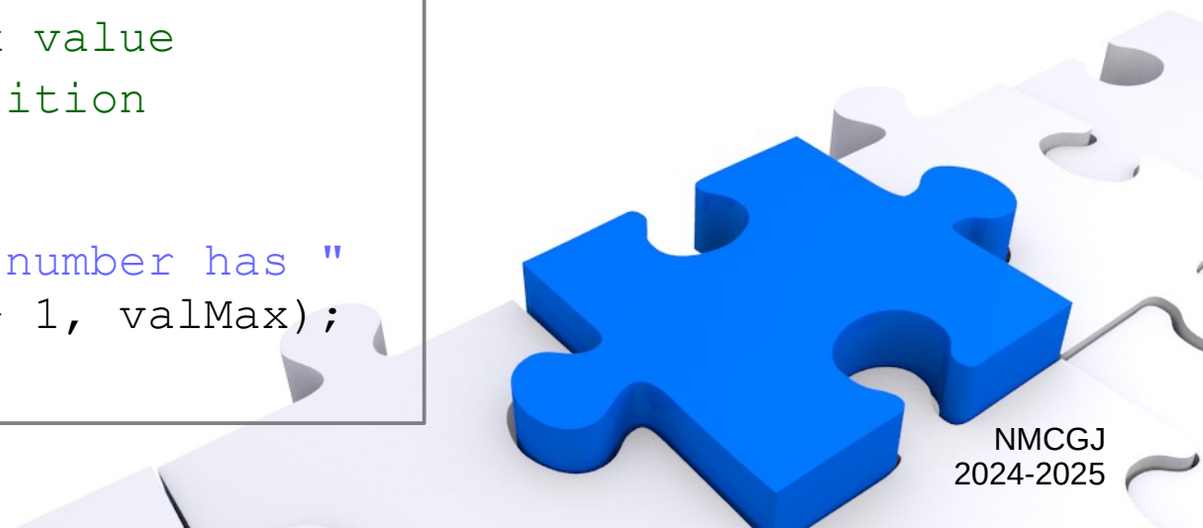
```
int[] list = {24, 14, 10, 30, 45, 2,
              1, 23,  5, 34, 23, 8};
```

Strings and Arrays

Università di Roma Tor Vergata

- Java program: arrays
  - Searching maximum element in an unsorted array

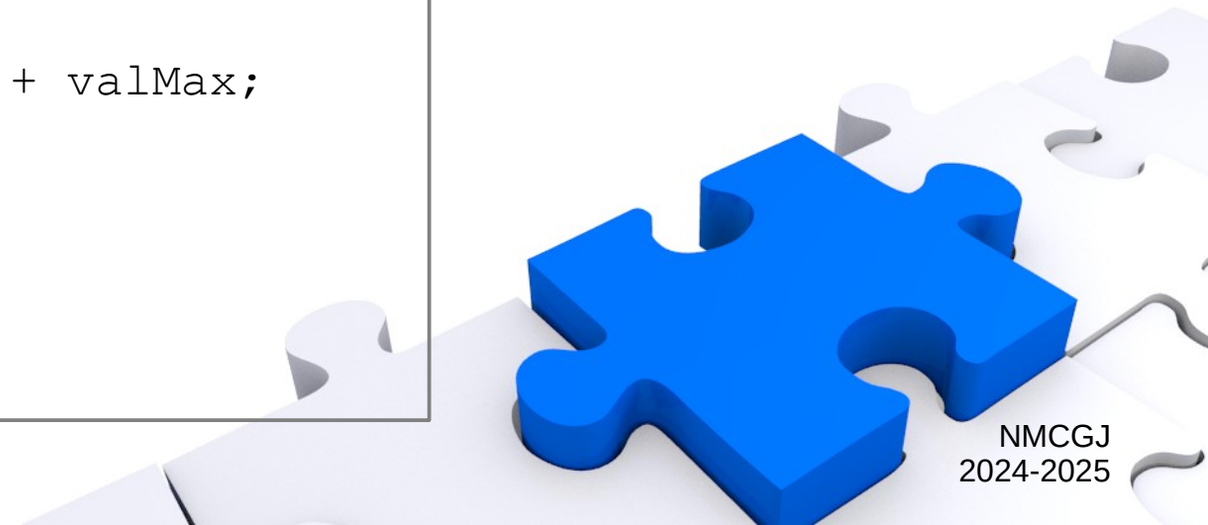
```
int valMax = list[0];
int posMax = 0;

for (int i = 1; i < list.length; i++)
   if (list[i] > valMax) {
      valMax = list[i]; // update max value
      posMax = i;       // update position
   }

String str = String.format("The %dth number has "
   + "the maximum value %d", posMax + 1, valMax);
System.out.println(str);
```



Strings and Arrays

- Java program: arrays
  - Searching maximum element in an unsorted array (bis)

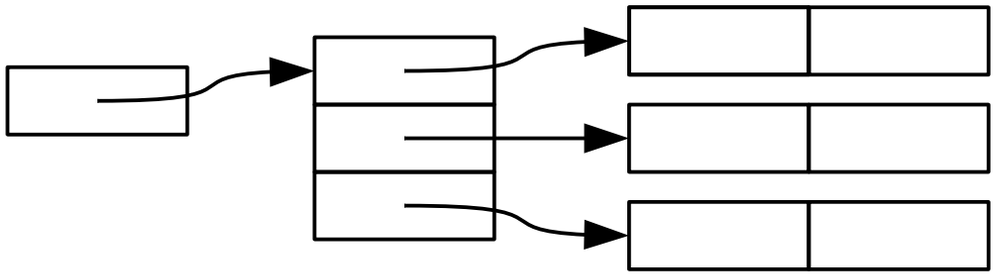
```
int valMax = list[0];

for (int val : list)
   if (val > valMax)
      valMax = val;      // update max value

String str = "The maximum value is " + valMax;
System.out.println(str);
```

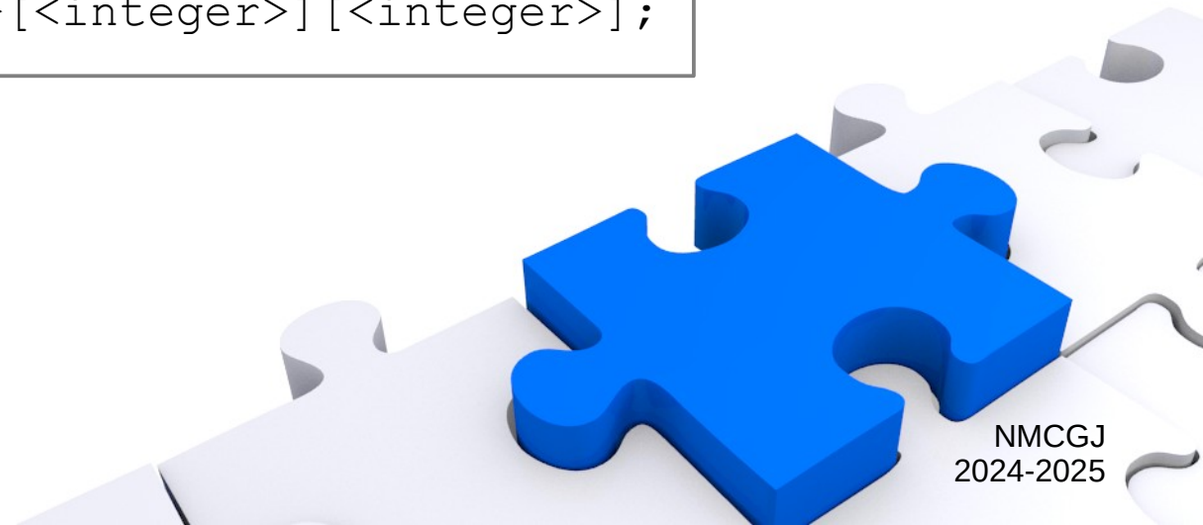


Strings and Arrays

- Two-dimensional arrays
  - A two-dimensional array is an array of arrays
    - Arrays are objects
    - Objects can be elements in an array
  - Creating a matrix

```
<type>[][] <matrix name> = new <type>[<integer>][<integer>];
```

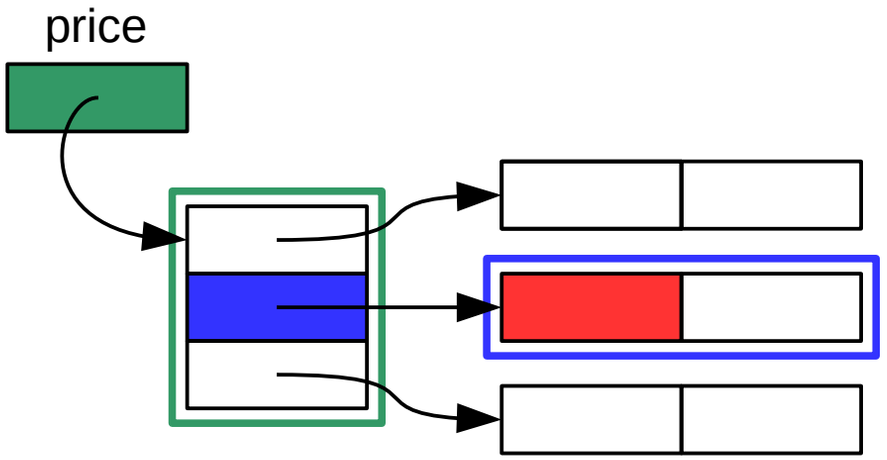




Strings and Arrays

- Two-dimensional arrays
  - Example

```
int[][] price = new int[3][2];
...
int value = price[1][0];
```



price

price[1]

price[1][0]

  - Number of rows and columns

```
int r = price.length;
int c = price[0].length;
```

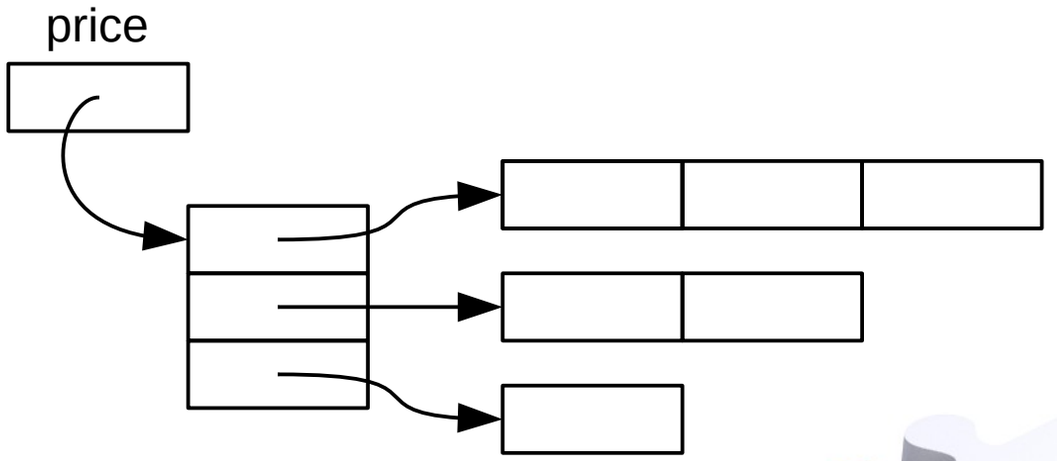
= 3

= 2

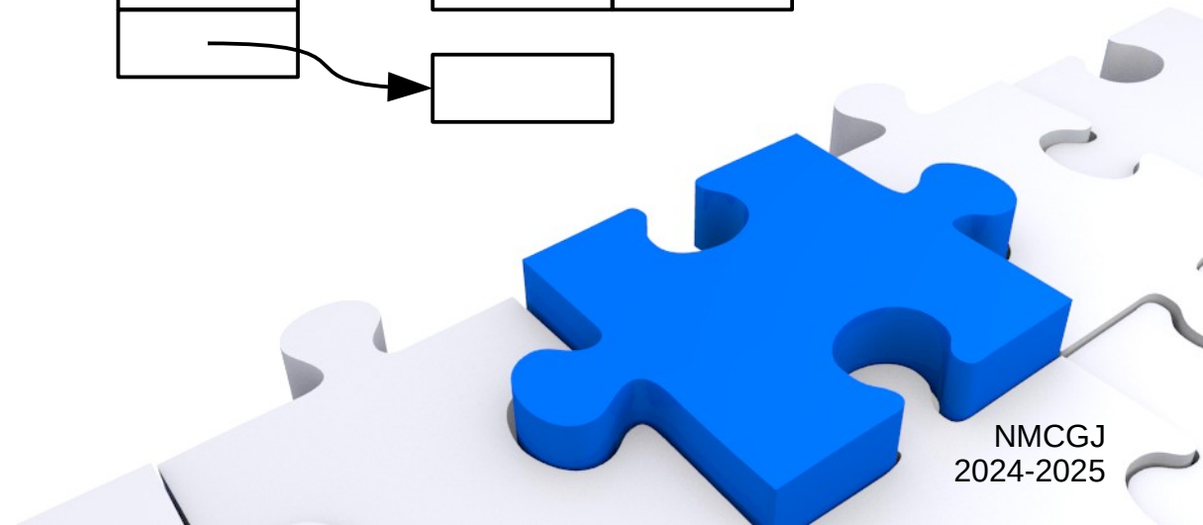
Strings and Arrays

- Two-dimensional arrays
  - A two-dimensional array is not always a matrix!

```
int[][] price = new int[3][];
price[0] = new int[3];
price[1] = new int[2];
price[2] = new int[1];
```



  - Multi-dimensional arrays
    - straightforward extension



Strings and Arrays

- Two-dimensional arrays
  - Example: printing a given matrix to the console

```
int[][] matrix = {{1,2,3},
                  {4,5,6}};

for (int i = 0; i < matrix.length; i++)
   for (int j = 0; j < matrix[i].length; j++)
      System.out.printf(
         "matrix[%d][%d] = %d\n",
         i, j, matrix[i][j]
      );
```

Strings and Arrays

Università di Roma Tor Vergata

- Two-dimensional arrays
  - Example: printing a given matrix to the console (bis)

```
int[][] matrix = {{1,2,3},
                  {4,5,6}};

System.out.println("matrix = ");
for (int[] row : matrix) {
   System.out.print("  [");
   for (int entry : row)
      System.out.printf("[%d]", entry);
   System.out.println("]");
}
```

Strings and Arrays

Università di Roma Tor Vergata

- Container classes
  - Arrays
    - Most efficient way to hold a group of objects (compiler supported)
    - Primitives or objects of same type
    - Fixed size
  - Container classes
    - Part of the library java.util
    - Sophisticated ways to hold your objects
      - Exploiting characteristics of your group (List, Set, Map, ...)
    - Objects of same type (based on the concept of generics)
    - Automatically resizing

Strings and Arrays

Università di Roma Tor Vergata

- Container classes
  - Overview of basic interfaces
    - **Collection**: a sequence of elements according to some rules
      - **List**: holds the elements in the way they were inserted
      - **Set**: cannot have duplicate elements
      - **Queue**: holds the elements in the order of a queuing discipline
    - **Map**: a group of key-value object pairs
      - Look up an object using another object (principle of dictionary)
    - These interfaces allow you to hold your objects without caring about the exact implementation

Strings and Arrays

Università di Roma Tor Vergata

- Container classes
  - Overview of basic interfaces
    - **Collection**: a sequence of elements according to some rules
      - **List**: holds the elements in the way they were inserted
      - **Set**: cannot have duplicate elements
      - **Queue**: holds the elements in the order of a queuing discipline
    - **Map**: a group of key-value object pairs
      - Look up an object using another object (principle of dictionary)
    - These interfaces allow you to hold your objects without caring about the exact implementation

Strings and Arrays

- Container classes
  - Implementations of the interface **List**
    - **ArrayList**: extends the classical array
      - Fast random access to elements
      - Expensive insertion / deletion of elements in the middle
    - **LinkedList**
      - Optimal sequential access, but slow random access
      - Inexpensive insertion / deletion of elements in the middle
  - Syntax of List creation

```
List<Apple> apples = new ArrayList<Apple>();
List<Orange> oranges = new LinkedList<Orange>();
```

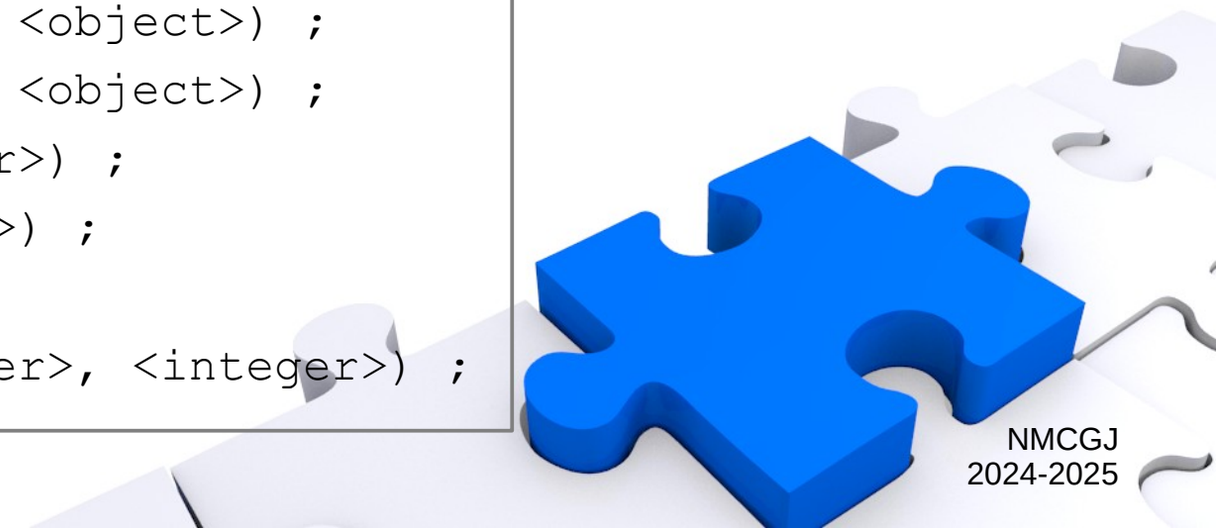
generics: specify type of objects to be held in List between angle brackets (since Java SE5)

Strings and Arrays

Università di Roma Tor Vergata

- Container classes
  - Some useful List methods

```
<integer> = <list> . size () ;
 <object> = <list> . get (<integer>) ;
<integer> = <list> . indexOf (<object>) ;
<boolean> = <list> . add (<object>) ;
            <list> . add (<integer>, <object>) ;
 <object> = <list> . set (<integer>, <object>) ;
 <object> = <list> . remove (<integer>) ;
<boolean> = <list> . remove (<object>) ;
            <list> . clear () ;
   <list> = <list> . subList (<integer>, <integer>) ;
```

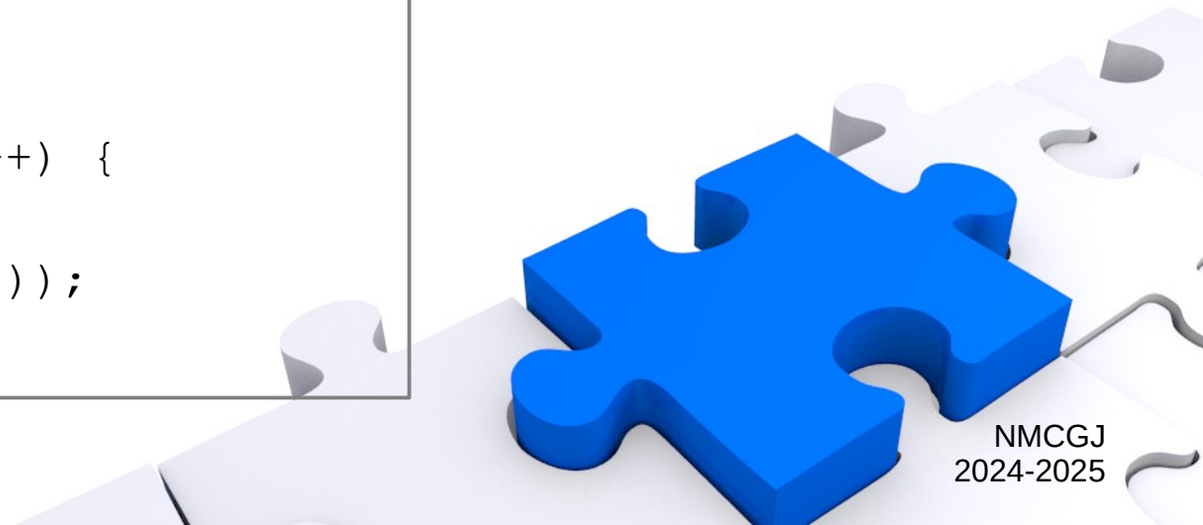


Strings and Arrays

- Container classes
  - Example: iterating over a List (using for loop)

```
class Apple {}
class Gala extends Apple {}
class Fuji extends Apple {}

ArrayList<Apple> apples = new ArrayList<Apple>();
apples.add(new Gala());
apples.add(new Fuji());
for (int i = 0; i < apples.size(); i++) {
   Apple apple = apples.get(i);
   System.out.println(apple.toString());
}
```

get() is fast for ArrayList
but slow for LinkedList



Strings and Arrays

Università di Roma Tor Vergata

- Container classes
  - The interface **Iterator** is designed for
    - moving efficiently through a sequence (Collection), and
    - selecting each object in the sequence without knowing its structure
  - Implementation depends on the specific container
  - Some useful Iterator methods

```
    <iterator> = <collection> . iterator () ;
<listiterator> = <list> . listIterator () ;
```

```
     <object> = <iterator> . next () ;
    <boolean> = <iterator> . hasNext () ;
```

Strings and Arrays

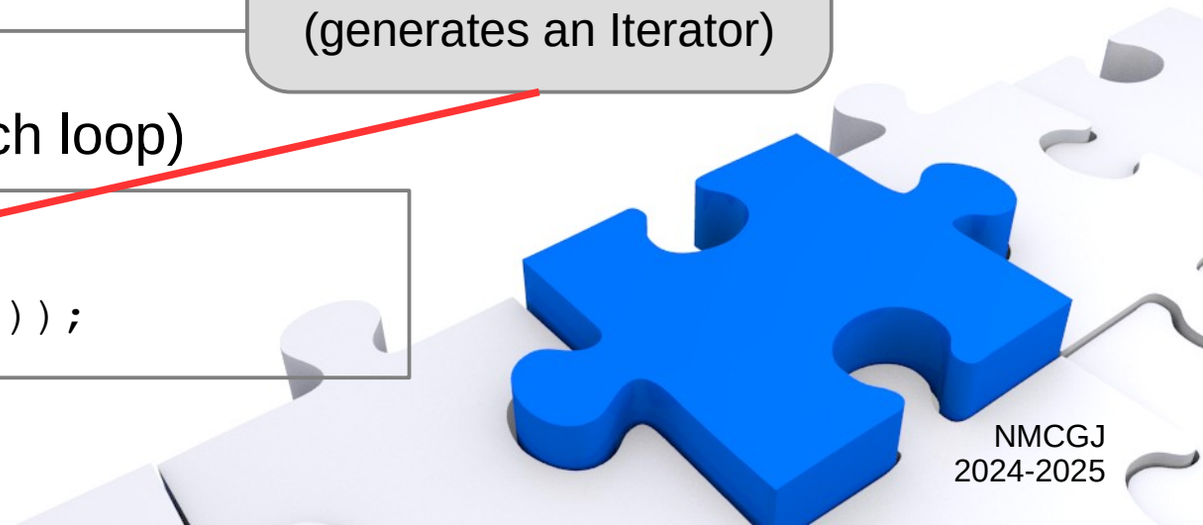
- Container classes
  - Example: iterating over a List (using an Iterator)

```
Iterator<Apple> iter = apples.iterator();
while (iter.hasNext()) {
    Apple apple = iter.next();
    System.out.println(apple.toString());
}
```

a **foreach** loop works with any **Iterable** object (generates an Iterator)

  - Example: iterating over a List (using foreach loop)

```
for (Apple apple : apples)
    System.out.println(apple.toString());
```



Strings and Arrays

- Container classes: overview