LECTURE 04: COLLECTION CLASSES

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

UNIVERSITY of GREENWICH
Alliance with FPT Education

PLAN

- Strings
- Regular Expressions
- Arrays
- Lists
- Maps

UNIVERSITY of GREENWICH
Alliance with FPT Education

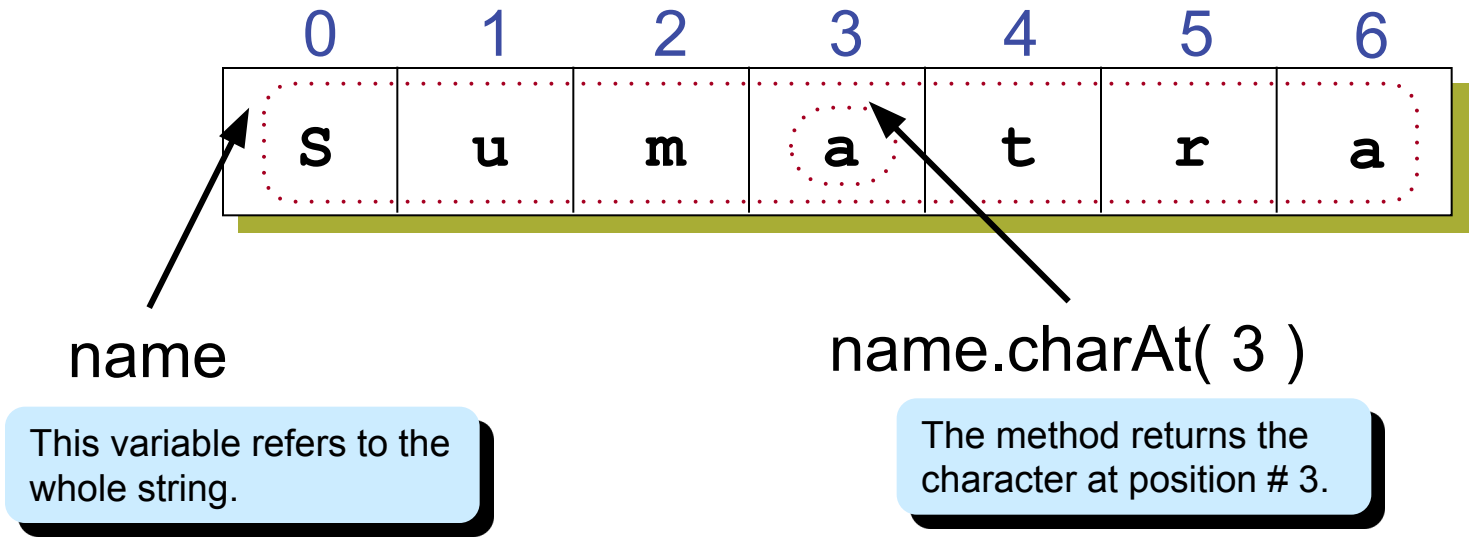
Strings

- A string is a sequence of characters that is treated as a single value.
- Instances of the String class are used to represent strings in Java.
- We can access individual characters of a string by calling the charAt method of the String object.

Accessing Individual Elements

- Individual characters in a String accessed with the charAt method.

```
String name = "Sumatra";
```



UNIVERSITY of GREENWICH
Alliance with FPT

Example: Counting Vowels

```
char     letter;
System.out.println ("Your name:");
String   name = scanner.next (); //assume 'scanner' is created properly
int      numberOfCharacters    = name.length ();
int      vowelCount        = 0;

for (int i = 0; i < numberOfCharacters; i++ ) {
    letter = name.charAt (i);

    if (     letter == 'a' || letter == 'A' ||
             letter == 'e' || letter == 'E' ||
             letter == 'i' || letter == 'I' ||
             letter == 'o' || letter == 'O' ||
             letter == 'u' || letter == 'U'
                           ) {
        vowelCount++;
    }
}
System.out.print (name + ", your name has " + vowelCount + " vowels");
```

Here's the code to count the number of vowels in the input string.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Example: Counting 'Java'

```
int           javaCount    = 0;
boolean       repeat           = true;
String        word;
Scanner       scanner =  new Scanner(System.in);

while  ( repeat )  {
     System.out.print ("Next word:");
     word = scanner.next ();

     if  ( word.equals ("STOP") )      {
          repeat = false;

     } else if  ( word.equalsIgnoreCase ("Java ) ) {
          javaCount++;
     }
}
```

Continue reading words and count how many times the word Java occurs in the input, ignoring the case.

Notice how the comparison is done. We are not using the == operator.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Other Useful String Operators

Method	Meaning
compareTo	Compares the two strings. `str1.compareTo( str2 )`
substring	Extracts the a substring from a string. `str1.substring( 1, 4 )`
trim	Removes the leading and trailing spaces. `str1.trim( )`
valueOf	Converts a given primitive data value to a string. `String.valueOf( 123.4565 )`
startsWith	Returns true if a string starts with a specified prefix string. `str1.startsWith( str2 )`
endsWith	Returns true if a string ends with a specified suffix string. `str1.endsWith( str2 )`

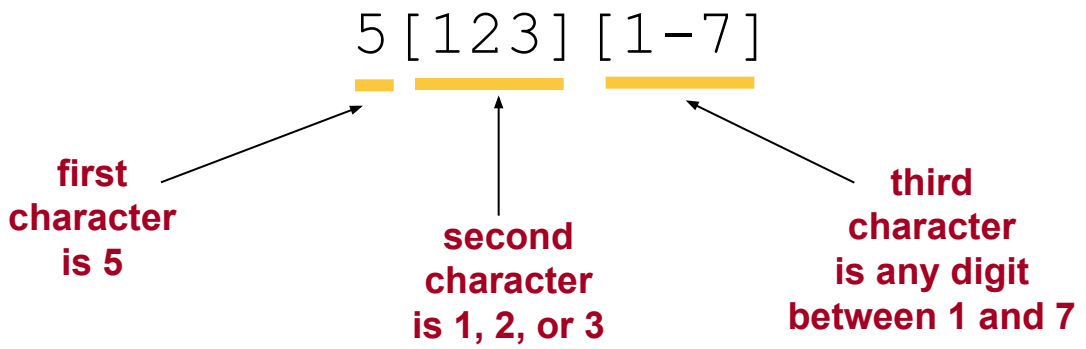
- See the String class documentation for details.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pattern Example

- Suppose students are assigned a three-digit code:
  - The first digit represents the major (5 indicates computer science);
  - The second digit represents either in-state (1), out-of-state (2), or foreign (3);
  - The third digit indicates campus housing:
    - On-campus dorms are numbered 1-7.
    - Students living off-campus are represented by the digit 8.

The 3-digit pattern to represent computer science majors living on-campus is



UNIVERSITY of GREENWICH
Alliance with FPT Education

Regular Expressions

- The pattern is called a regular expression.
- Rules
  - The brackets [ ] represent choices
  - The asterisk symbol * means zero or more occurrences.
  - The plus symbol + means one or more occurrences.
  - The hat symbol ^ means negation.
  - The hyphen – means ranges.
  - The parentheses ( ) and the vertical bar | mean a range of choices for multiple characters.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Regular Expression Examples

Expression	Description
`[013]`	A single digit 0, 1, or 3.
`[0-9][0-9]`	Any two-digit number from 00 to 99.
`[0-9&&[^4567]]`	A single digit that is 0, 1, 2, 3, 8, or 9.
`[a-z0-9]`	A single character that is either a lowercase letter or a digit.
`[a-zA-Z_][a-zA-Z0-9_$]*`	A valid Java identifier consisting of alphanumeric characters, underscores, and dollar signs, with the first character being an alphabet.
`[wb](ad|eed)`	Matches `wad`, `weed`, `bad`, and `beed`.
`(AZ|CA|CO)[0-9][0-9]`	Matches `AZxx`,`CAxx`, and `COxx`, where `x` is a single digit.

UNIVERSITY of GREENWICH
Alliance with FPT Education

The replaceAll Method

- The replaceAll method replaces all occurrences of a substring that matches a given regular expression with a given replacement string.

Replace all vowels with the symbol @

```
String originalText, modifiedText;

originalText = ...;     //assign string

modifiedText =
          originalText.replaceAll("[aeiou]","@");
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

The Pattern and Matcher Classes

- The matches and replaceAll methods of the String class are shorthand for using the Pattern and Matcher classes from the java.util.regex package.
- If str and regex are String objects, then

```
str.matches(regex);
```

is equivalent to

```
Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(str);
matcher.matches();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

The compile Method

- The compile method of the Pattern class converts the stated regular expression to an internal format to carry out the pattern-matching operation.
- This conversion is carried out every time the matches method of the String class is executed, so it is more efficient to use the compile method when we search for the same pattern multiple times.

UNIVERSITY of GREENWICH
Alliance with FPT Education

The find Method

- The find method is another powerful method of the Matcher class.
  - It searches for the next sequence in a string that matches the pattern, and returns true if the pattern is found.
- When a matcher finds a matching sequence of characters, we can query the location of the sequence by using the start and end methods.

UNIVERSITY of GREENWICH
Alliance with FPT Education

The String Class is Immutable

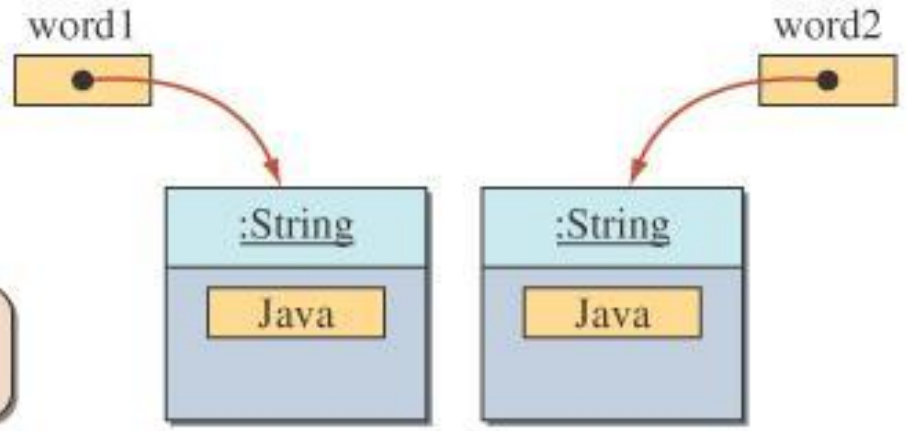
- In Java a String object is immutable
  - This means once a String object is created, it cannot be changed, such as replacing a character with another character or removing a character
  - The String methods we have used so far do not change the original string. They created a new string from the original. For example, substring creates a new string from a given string.
- The String class is defined in this manner for efficiency reason.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Effect of Immutability

```
String word1, word2;

word1 = new String("Java");

word2 = new String("Java");
```

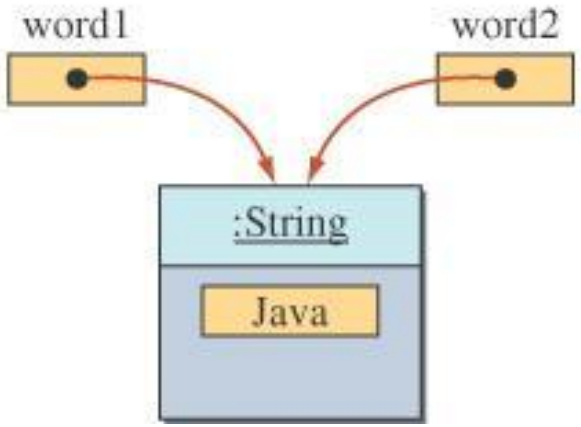
Whenever the **new** operator is used, there will be a new object.




```
String word1, word2;

word1 = "Java";

word2 = "Java";
```

Literal string constant such as "Java" will always refer to the one object.

We can do this because String objects are immutable.



UNIVERSITY of GREENWICH
Alliance with FPT Education

The StringBuffer Class

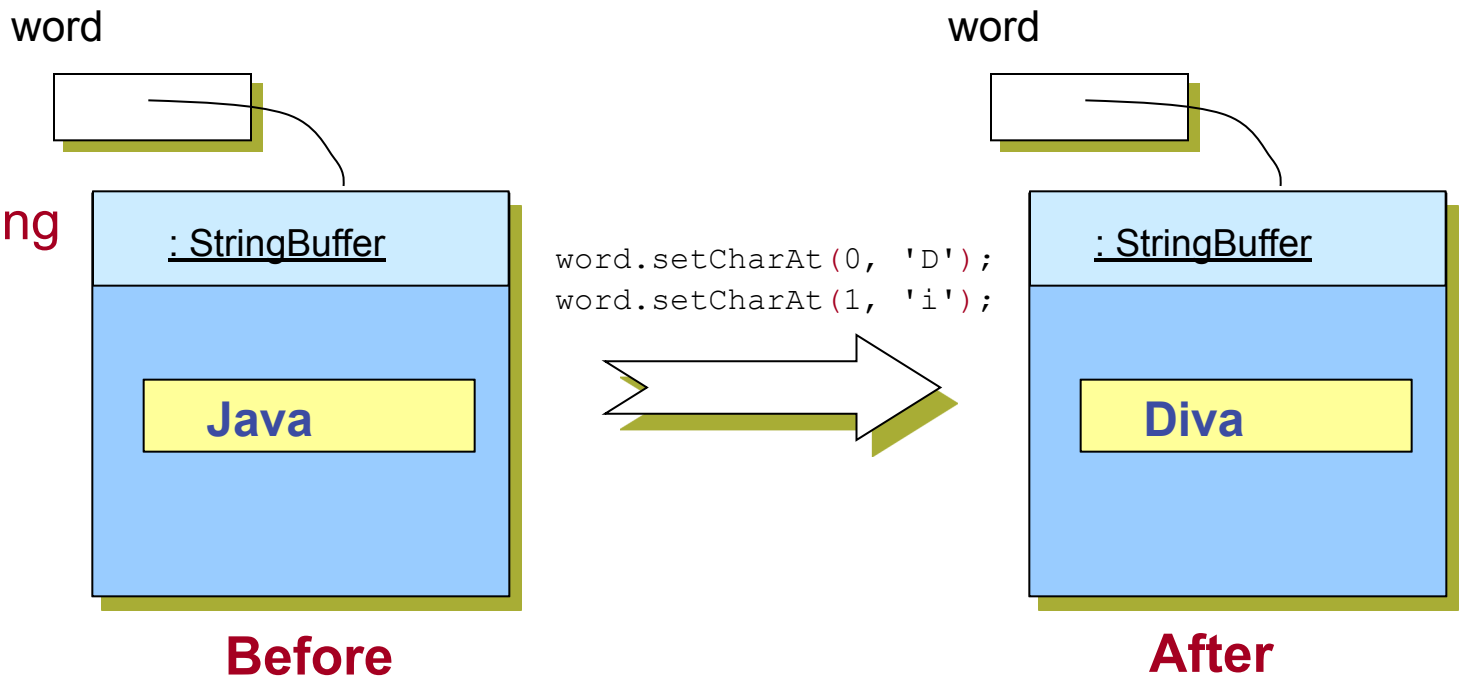
- In many string processing applications, we would like to change the contents of a string. In other words, we want it to be mutable.
- Manipulating the content of a string, such as replacing a character, appending a string with another string, deleting a portion of a string, and so on, may be accomplished by using the StringBuffer class.

UNIVERSITY of GREENWICH
Alliance with FPT Education

StringBuffer Example

```
StringBuffer word = new StringBuffer("Java");
word.setCharAt(0, 'D');
word.setCharAt(1, 'i');
```

Changing a string
Java to Diva



UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample Processing

- Replace all vowels in the sentence with ‘X’.

```
char          letter;
String        inSentence    = scanner.nextLine();
StringBuffer tempStringBuffer    =  new StringBuffer(inSentence);
int           numberOfCharacters = tempStringBuffer.length ();

for (int index = 0; index < numberOfCharacters; index++ ) {

    letter = tempStringBuffer.charAt (index);

    if (letter == 'a' || letter == 'A' || letter == 'e' || letter == 'E'
||
        letter == 'i' || letter == 'I' || letter == 'o' || letter == 'O'
||
        letter == 'u' || letter == 'U'   ) {
        tempStringBuffer.setCharAt (index,'X');
    }
}
System.out.println(tempStringBuffer  );
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

The append and insert Methods

- We use the append method to append a String or StringBuffer object to the end of a StringBuffer object.
  - The method can also take an argument of the primitive data type.
  - Any primitive data type argument is converted to a string before it is appended to a StringBuffer object.
- We can insert a string at a specified position by using the insert method.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Array vs Object

- In Java, arrays are actually object. There is a class for any array type but programmer doesn’t need to know it exactly.
- In Java, in addition to arrays of primitive data types, we can declare arrays of objects
- The use of an array of objects allows us to model the application more cleanly and logically.

Creating Array of Objects

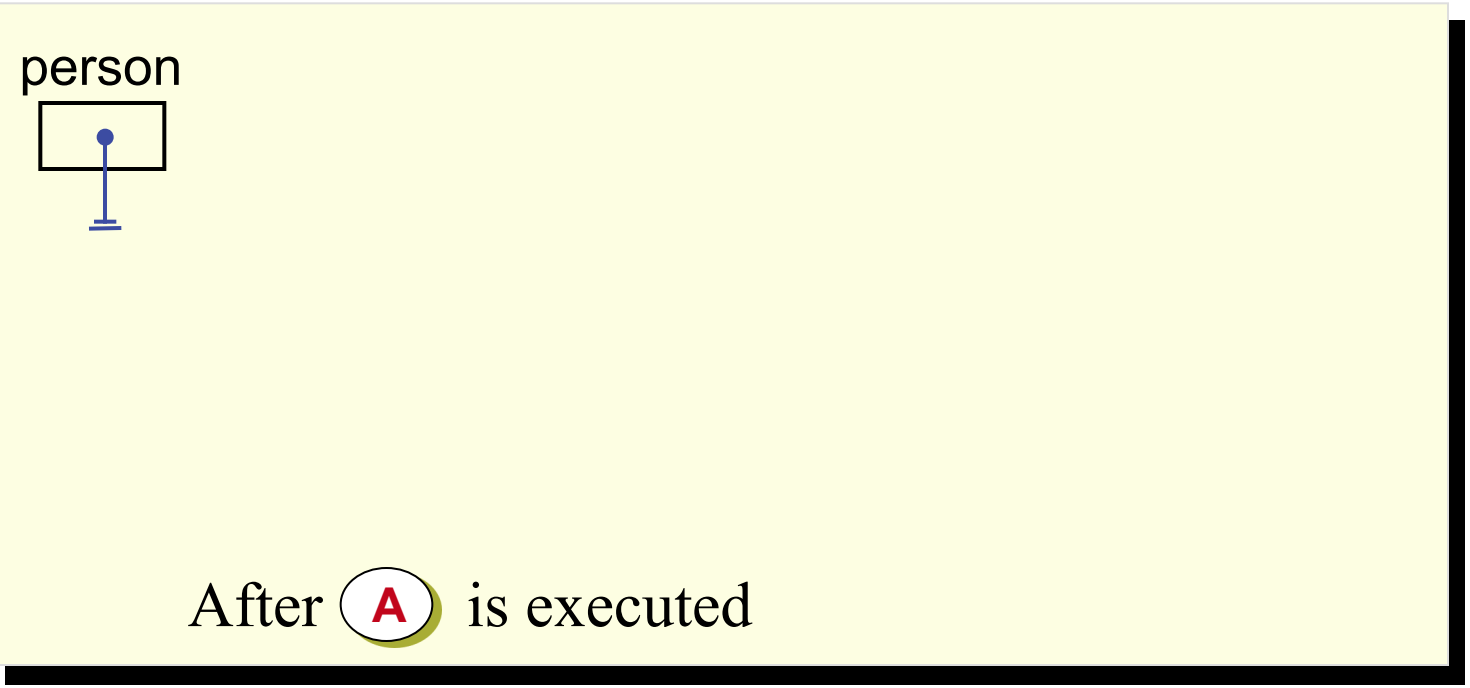
Code

A

```
Person[ ]  person;
person = new Person[20];
person[0] = new Person( );
```

Only the name person is declared, no array is allocated yet.

State of Memory



Creating Array of Objects

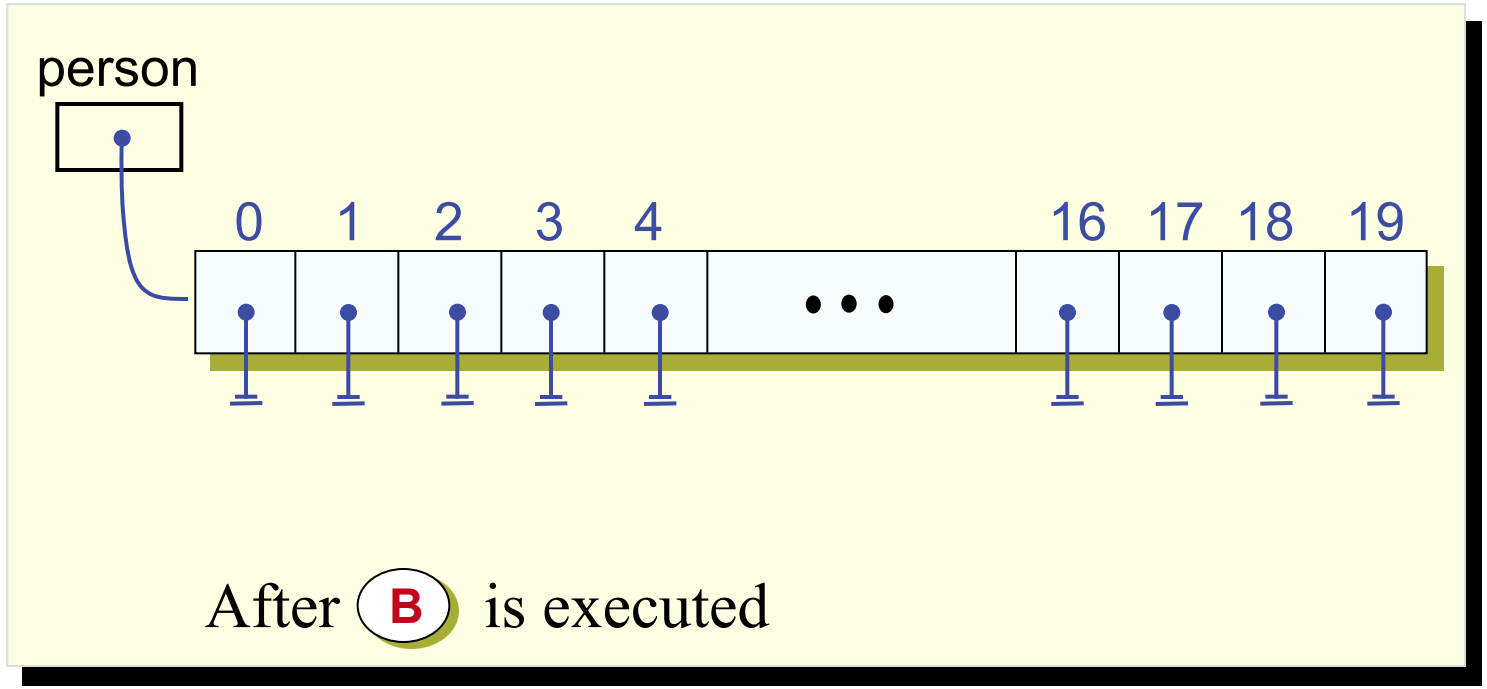
Code

B

```
Person[ ]  person;
person = new Person[20];
person[0] = new Person( );
```

Now the array for storing 20 Person objects is created, but the Person objects themselves are not yet created.

State of Memory



Creating Array of Objects

UNIVERSITY of GREENWICH
Alliance with FPT Education

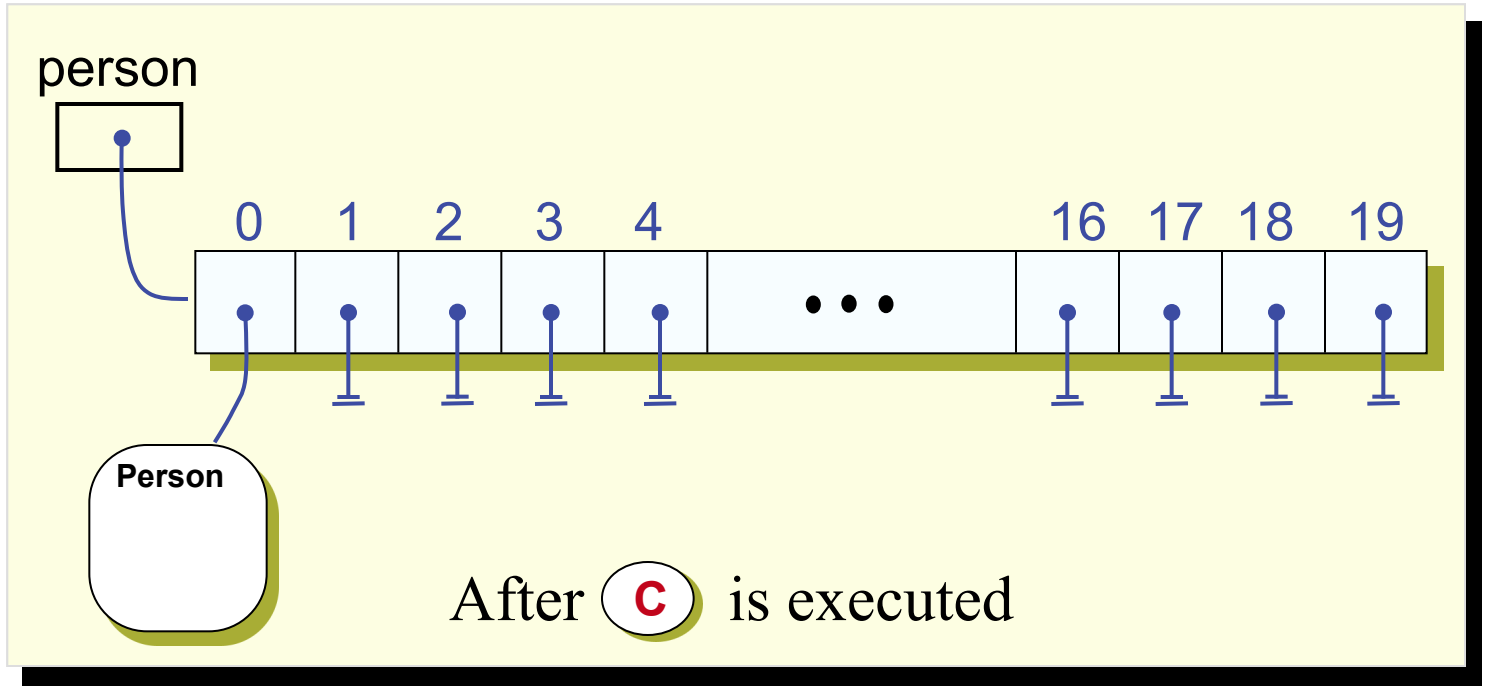
Code

```
Person[ ]  person;
person = new Person[20];
person[0] = new Person( );
```

C

One Person object is created and the reference to this object is placed in position 0.

State of Memory



UNIVERSITY of GREENWICH
Alliance with FPT Education

Person Array Processing – Sample 1

```
String       name, inpStr; int age; char gender;
Scanner scanner = new Scanner(System.in);

for (int i = 0; i < person.length; i++ ) {
    System.out.print("Enter name:"); name = scanner.next ( );
    System.out.print("Enter age:");  age = scanner.nextInt( );
    System.out.print("Enter gender:"); inpStr = scanner.next( );
    gender = inpStr.charAt (0);

    person[i] = new Person( );     //create a new Person and assign value s

    person[i].setName   ( name    );
    person[i].setAge    (  age    );
    person[i].setGender ( gender );
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Person Array Processing – Sample 2

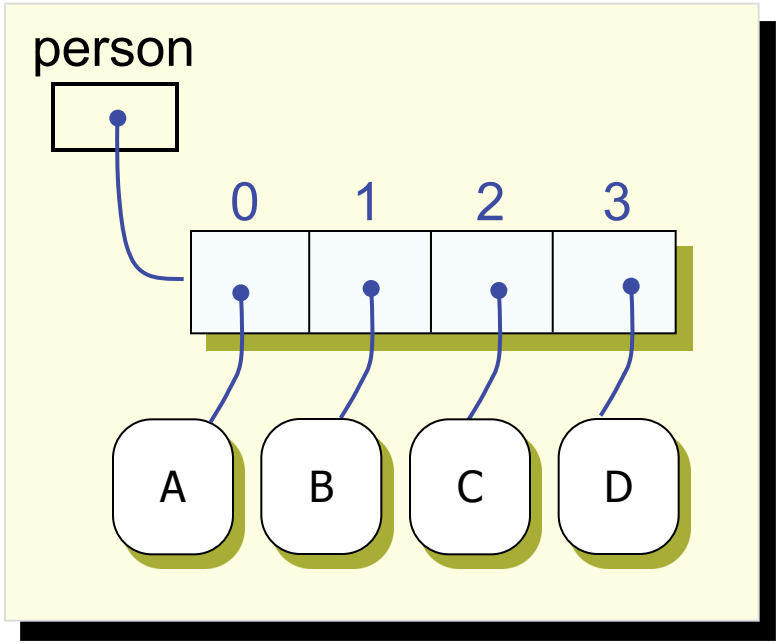
```
int      minIdx = 0;       //index to the youngest person
int      maxIdx = 0;       //index to the oldest person

for (int i = 1; i < person.length; i++ ) {

    if ( person[i].getAge() < person[minIdx].getAge() ) {
        minIdx           = i;     //found a younger person

    } else if (person[i].getAge() > person[maxIdx].getAge() ) {

        maxIdx            = i;    //found an older person
    }
}

//person[minIdx] is the youngest and person[maxIdx] is the oldest
```

Object Deletion – Approach 1

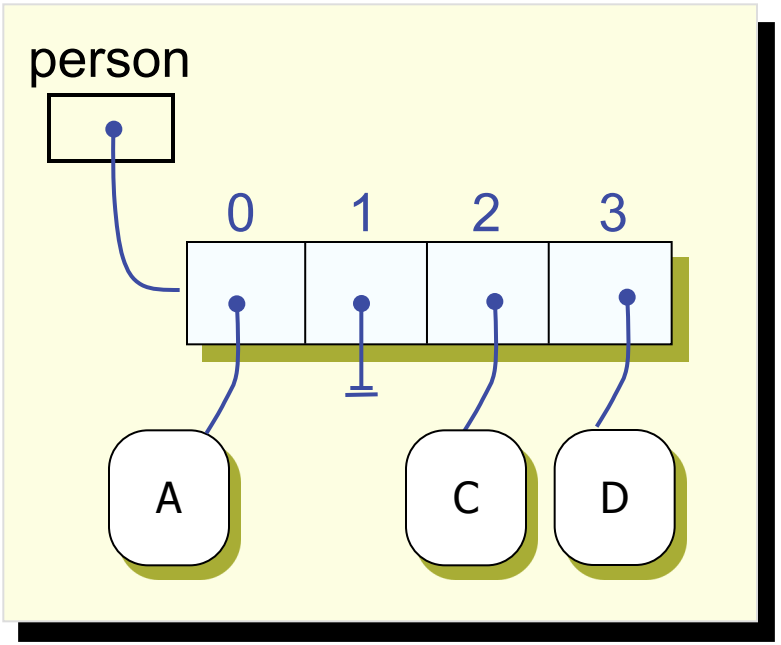
```
int delIdx = 1;

person[delIdx] = null;
```

A

Delete Person B by setting the reference in position 1 to null.



Before A is executed



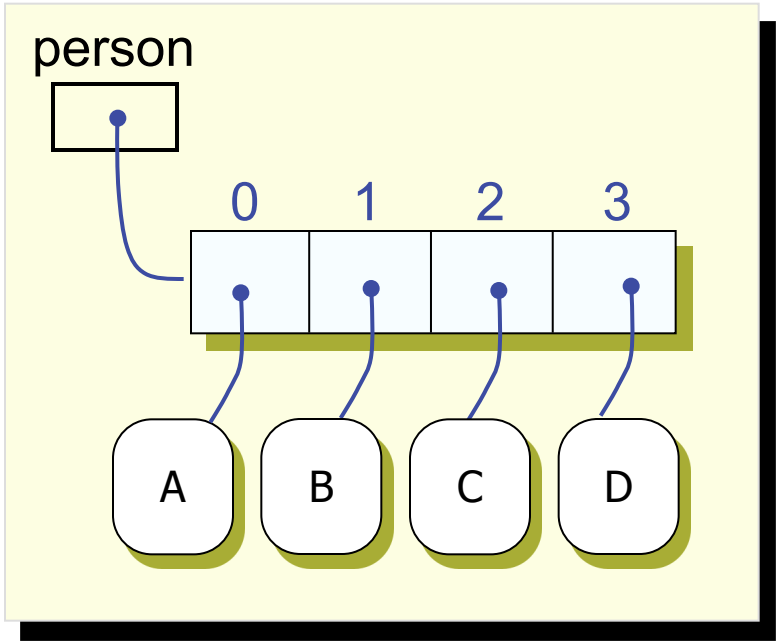
After A is executed

Object Deletion – Approach 2

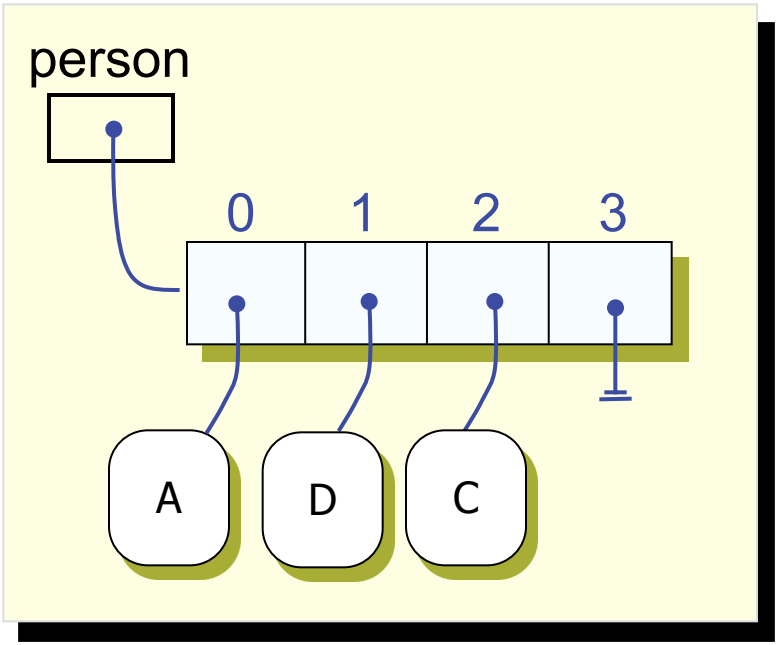
```
int delIdx = 1, last = 3;
person[delIndex] = person[last];
person[last]     = null;
```

A

Delete Person B by setting the reference in position 1 to the last person.



Before A is executed



After A is executed

UNIVERSITY of GREENWICH
Alliance with FPT Education

Person Array Processing – Sample 3

```
int i = 0;

while ( person[i] != null && !person[i].getName().equals("Latte") ) {
     i++;
}

if ( person[i] == null ) {
     //not found - unsuccessful search
     System.out.println("Ms. Latte was not in the array");

} else {
     //found - successful search
      System.out.println("Found Ms. Latte at position "  + i);
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

The For-Each Loop

- This new for loop is available from Java 5.0
- The for-each loop simplifies the processing of elements in a collection
- Here we show examples of processing elements in an array

```
int sum = 0;

for (int i = 0; i < number.length; i++ )
{
     sum = sum + number[i];
}
```

standard for loop

```
int sum = 0;

for (int value : number) {
     sum = sum + value;
}
```

for-each loop

UNIVERSITY of GREENWICH
Alliance with FPT Education

Processing an Array of Objects with For-Each

```
Person[] person = new Person[100];
//create person[0] to person[99]
```

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

standard for loop

```
for (Person p : person) {
    System.out.println(p.getName ());
}
```

for-each loop

UNIVERSITY of GREENWICH
Alliance with FPT Education

For-Each: Key Points to Remember

- A for-each loop supports read access only. The elements cannot be changed.
- A single for-each loop allows access to a single array only, i.e., you cannot access multiple arrays with a single for-each loop.
- A for-each loop iterates over every element of a collection from the first to the last element. You cannot skip elements or iterate backward.

Passing Arrays to Methods - 1

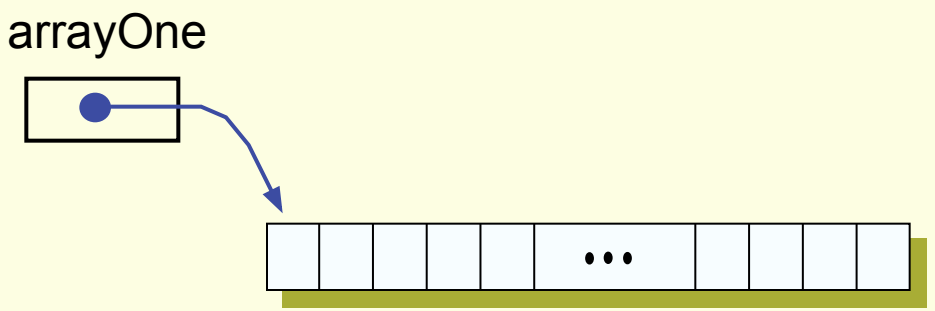
Code

A

```
minOne
= searchMinimum(arrayOne);
```

```
public int searchMinimum(float[] number))
{

      …


}
```

At A before searchMinimum



State of Memory

A. Local variable number does not exist before the method execution

Passing Arrays to Methods - 2

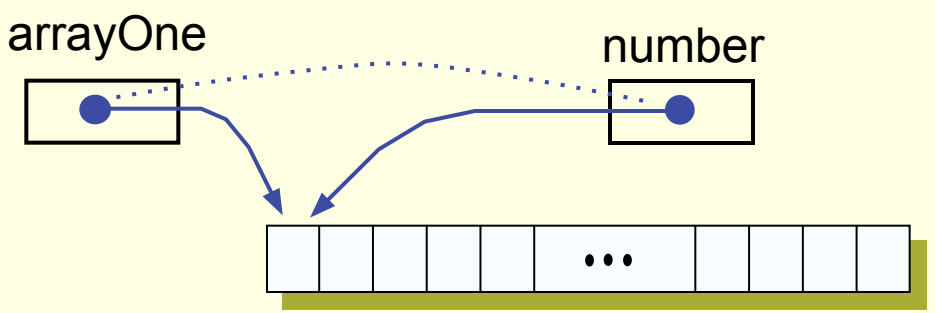
Code

```
minOne
= searchMinimum(arrayOne);
```

```
public int searchMinimum(float[] number))
{
      …


}
```

B

The address is copied at B



State of Memory

B. The value of the argument, which is an address, is copied to the parameter.

Passing Arrays to Methods - 3

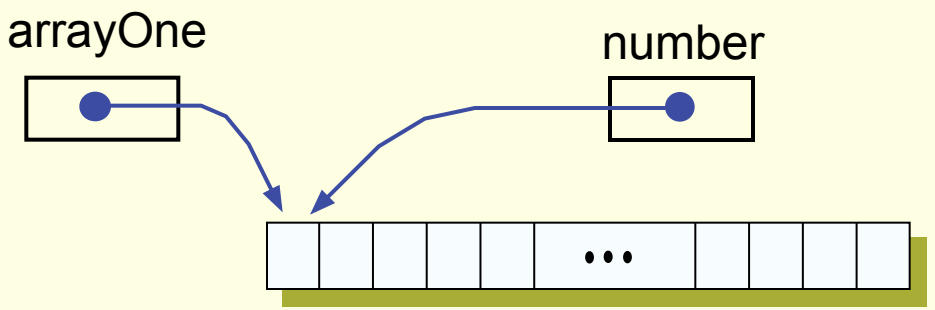
Code

```
minOne
= searchMinimum(arrayOne);
```

```
public int searchMinimum(float[] number))
{
      …
                                    C
}
```

State of Memory

While at C inside the method



C. The array is accessed via number inside the method.

Passing Arrays to Methods - 4

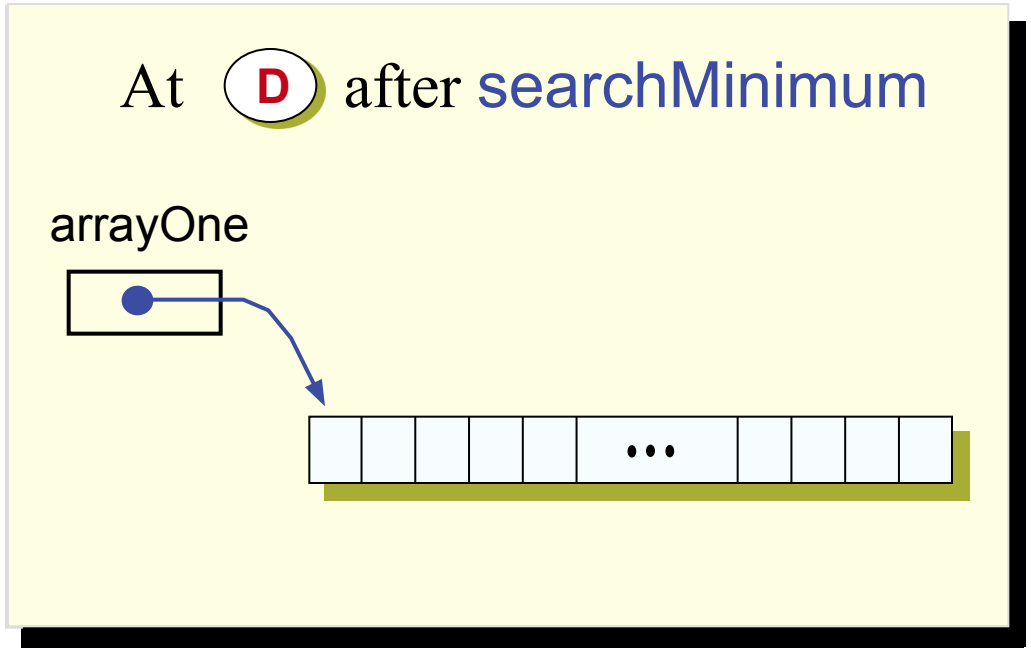
Code

```
minOne
= searchMinimum(arrayOne);
```

D

```
public int searchMinimum(float[] number))
{
      …
}
```

State of Memory



D. The parameter is erased. The argument still points to the same object.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Collection Classes: Lists and Maps

- The java.util standard package contains different types of classes for maintaining a collection of objects.
- These classes are collectively referred to as the Java Collection Framework (JCF).
- JCF includes classes that maintain collections of objects as sets, lists, or maps.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Java Interface

- A Java interface defines only the behavior of objects
  - It includes only public methods with no method bodies.
  - It does not include any data members except public constants
  - No instances of a Java interface can be created

UNIVERSITY of GREENWICH
Alliance with FPT Education

JCF Lists

- JCF includes the List interface that supports methods to maintain a collection of objects as a linear list

$$L = (l_0, l_1, l_2, \ldots, l_N)$$

- We can add to, remove from, and retrieve objects in a given list.
- A list does not have a set limit to the number of objects we can add to it.

UNIVERSITY of GREENWICH
Alliance with FPT Education

List Methods

`boolean add(E o)`
Adds an object o to the list
`void clear()`
Clears this list, i.e., make the list empty
`E get(int idx)`
Returns the element at position idx
`boolean remove(int idx)`
Removes the element at position idx
`int size ()`
Returns the number of elements in the list

E is a generic class.
Replace E with a concrete class.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Using Lists

- To use a list in a program, we must create an instance of a class that implements the List interface.
- Two classes that implement the List interface:
  - **ArrayList**
  - **LinkedList**

- The ArrayList class uses an array to manage data.
- The LinkedList class uses a technique called linked-node representation.

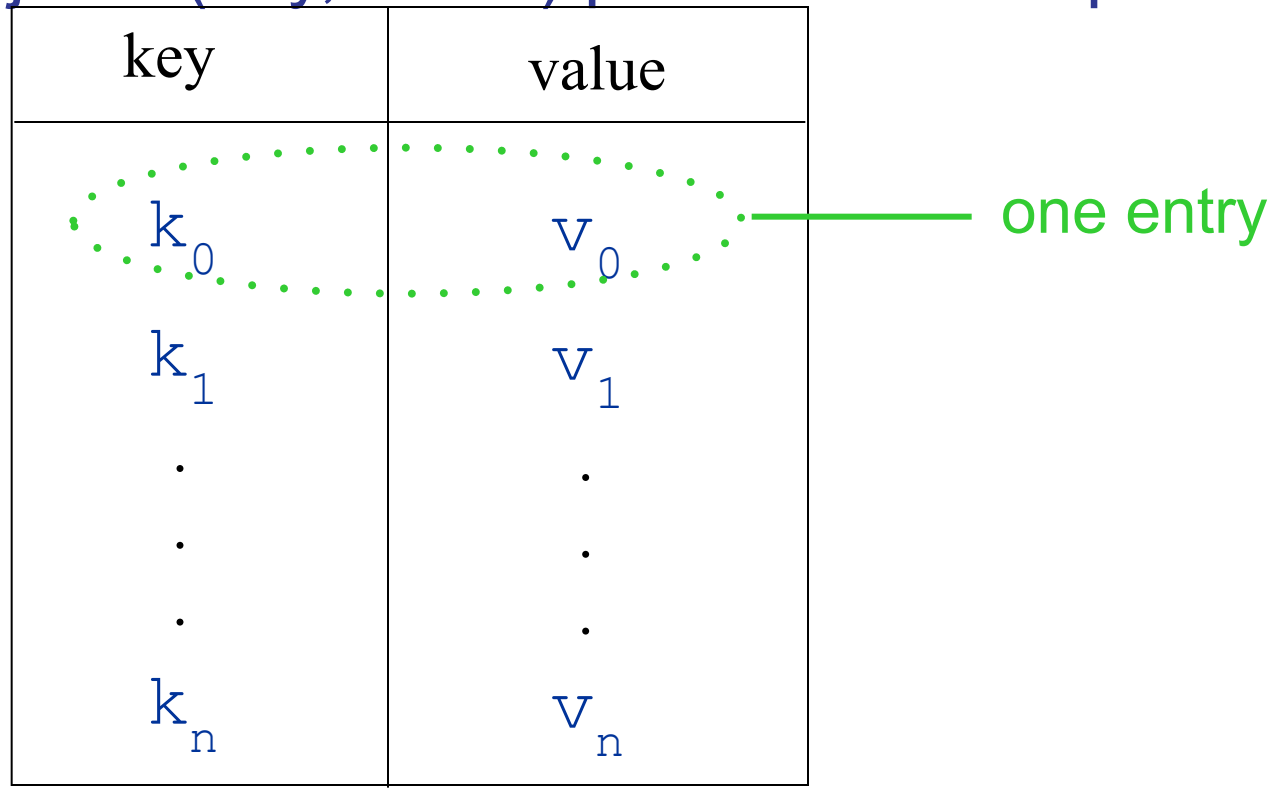
UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample List Usage

```
import java.util.*;

List<Person>  friends;
Person        person;

friends = new ArrayList<Person>( );

person = new Person("jane", 10, 'F');
friends.add( person );
person = new Person("jack",  6, 'M');
friends.add( person );

Person p = friends.get( 1 );
```

JCF Maps

- JCF includes the Map interface that supports methods to maintain a collection of objects (key, value) pairs called map entries.



UNIVERSITY of GREENWICH
Alliance with FPT Education

Map Methods

`void clear()`
Clears this list, i.e., make the map empty
`boolean containsKey(Object key)`
Returns true if the map contains an entry with a given key
`V put(K key, V value)`
Adds the given (key, value) entry to the map
`V remove(Object key)`
Removes the entry with the given key from the map
`int size()`
Returns the number of elements in the map

UNIVERSITY of GREENWICH
Alliance with FPT Education

Using Maps

- To use a map in a program, we must create an instance of a class that implements the Map interface.
- Two classes that implement the Map interface:
  - **HashMap**
  - **TreeMap**

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample Map Usage

```
import java.util.*;

Map     catalog;
catalog = new TreeMap<String, String>( );

catalog.put("CS101", "Intro Java Programming");
catalog.put("CS301", "Database Design");
catalog.put("CS413", "Software Design for Mobile Devices");

if (catalog.containsKey("CS101")) {
    System.out.println("We teach Java this semester");
} else {
    System.out.println("No Java courses this semester");
}
```