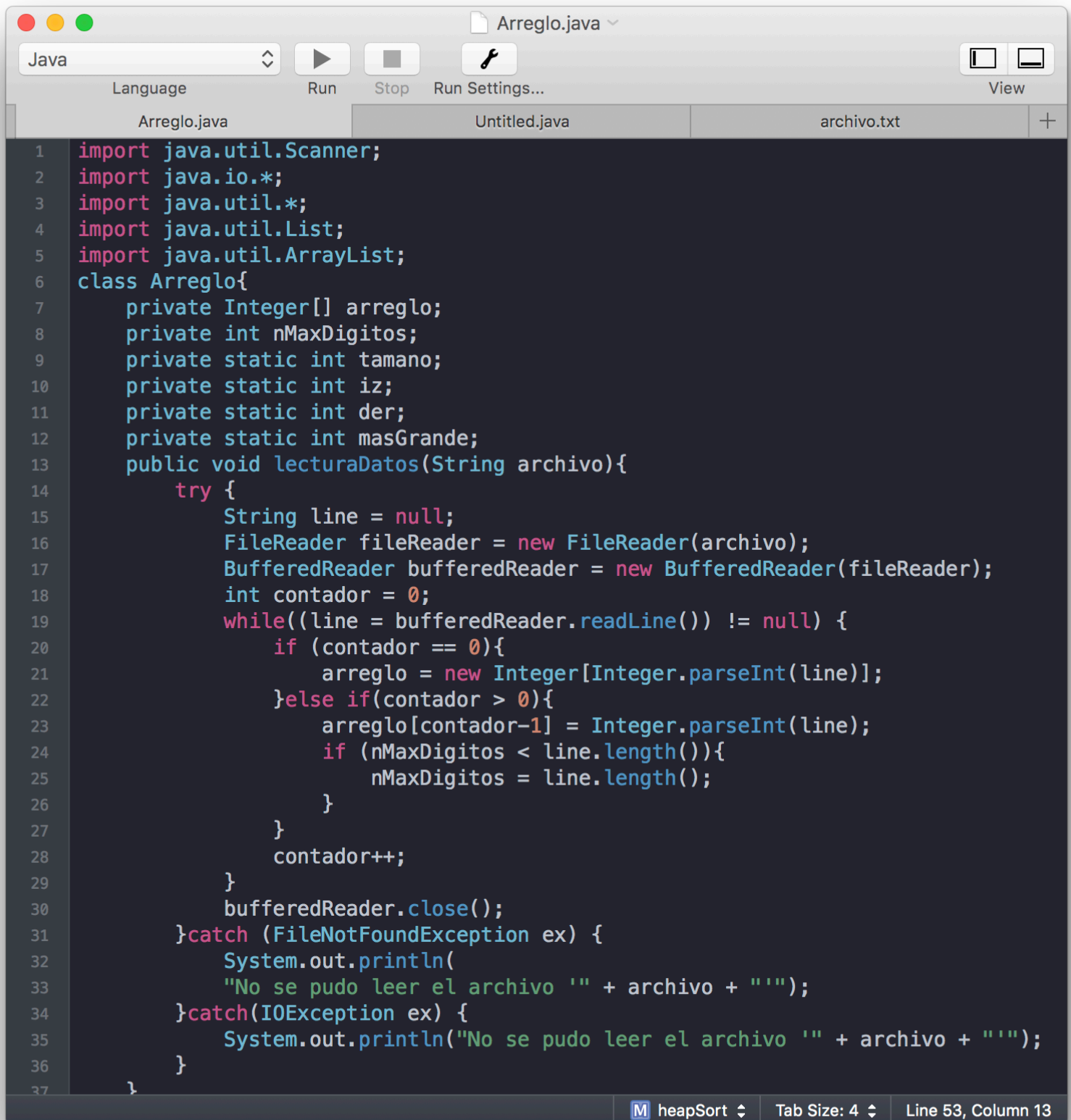


Documentación Proyecto 1

Emiliano Abascal Gurría

A01023234

Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Santa Fe



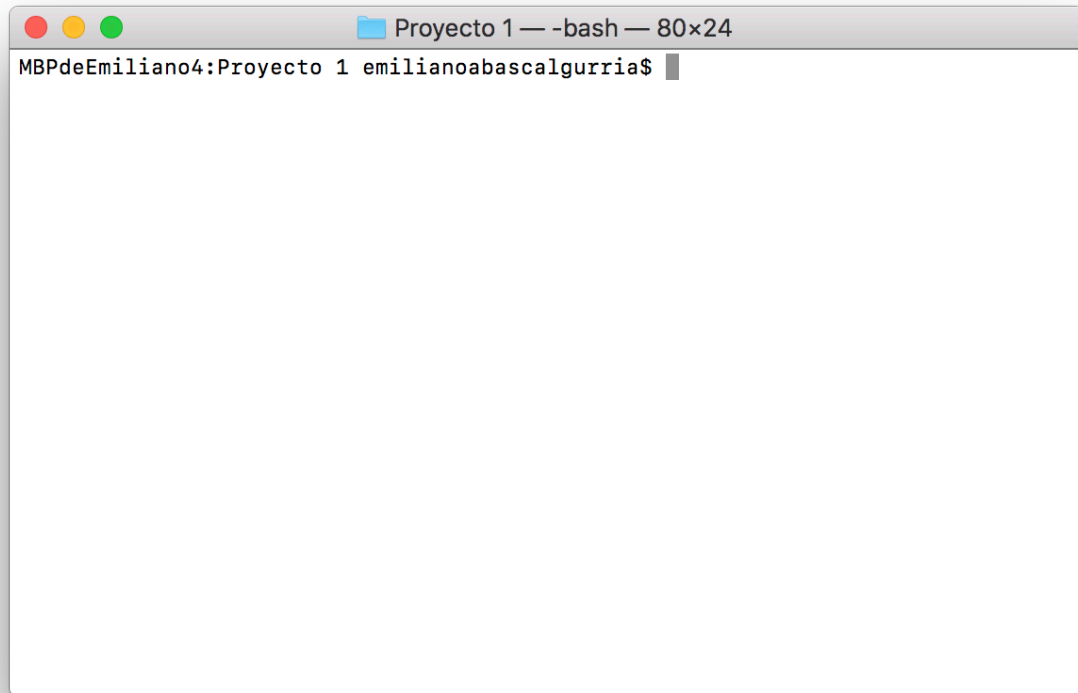
The screenshot shows an IDE window titled 'Arreglo.java'. The code is written in Java and defines a class 'Arreglo' with several attributes and a method 'lecturaDatos'. The code is as follows:

```
1 import java.util.Scanner;
2 import java.io.*;
3 import java.util.*;
4 import java.util.List;
5 import java.util.ArrayList;
6 class Arreglo{
7     private Integer[] arreglo;
8     private int nMaxDigitos;
9     private static int tamano;
10    private static int iz;
11    private static int der;
12    private static int masGrande;
13    public void lecturaDatos(String archivo){
14        try {
15            String line = null;
16            FileReader fileReader = new FileReader(archivo);
17            BufferedReader bufferedReader = new BufferedReader(fileReader);
18            int contador = 0;
19            while((line = bufferedReader.readLine()) != null) {
20                if (contador == 0){
21                    arreglo = new Integer[Integer.parseInt(line)];
22                }else if(contador > 0){
23                    arreglo[contador-1] = Integer.parseInt(line);
24                    if (nMaxDigitos < line.length()){
25                        nMaxDigitos = line.length();
26                    }
27                }
28                contador++;
29            }
30            bufferedReader.close();
31        }catch (FileNotFoundException ex) {
32            System.out.println(
33                "No se pudo leer el archivo '" + archivo + "'");
34        }catch (IOException ex) {
35            System.out.println("No se pudo leer el archivo '" + archivo + "'");
36        }
37    }
38 }
```

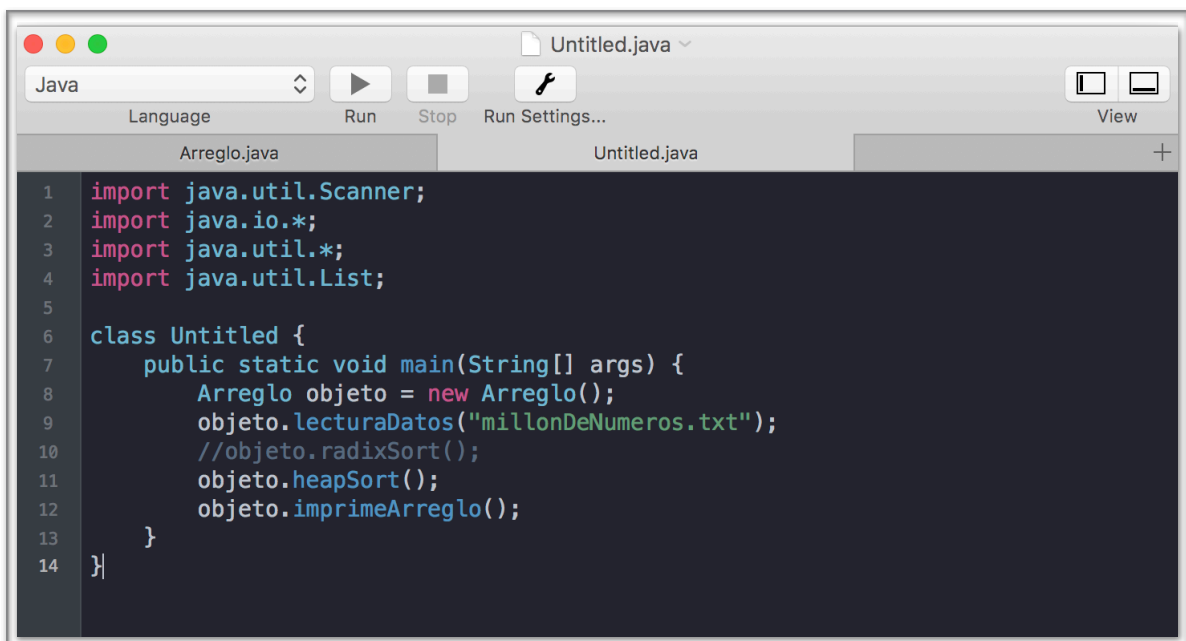
The IDE interface includes a toolbar with 'Run', 'Stop', and 'Run Settings...' buttons, and a 'View' button. The status bar at the bottom shows 'M heapSort', 'Tab Size: 4', and 'Line 53, Column 13'.

Manual de usuario:

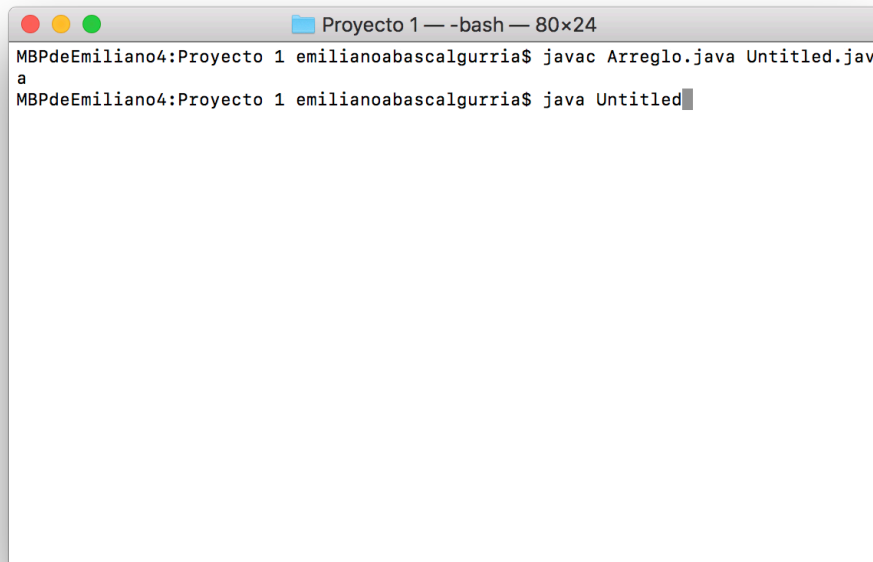
- Una vez obtenido el software ubicarlo en un directorio de preferencia, por ejemplo en el escritorio.
- Abrir la terminal o un compilador para correr programas hechos en Java.
- Escribir el siguiente comando: `cd "El directorio donde se encuentra el programa."`



- Ya que se está en el directorio se requiere hacer una clase de Java en el cual se hace una instancia de Arreglo, como se muestra a continuación:



- Al tener un archivo java se regresa a la terminal y escribe el siguiente comando: `javac Arreglo.java Untitled.java` y da enter.
- Escribe en la terminal `./a.out` y el programa se correra, como se demuestra en al ejemplo a continuación.



```

Proyecto 1 — -bash — 80x24
MBPdeEmiliano4:Proyecto 1 emilianoabascalgurria$ javac Arreglo.java Untitled.java
a
MBPdeEmiliano4:Proyecto 1 emilianoabascalgurria$ java Untitled

```

El programa fue probado con un archivo con 10 millones de numeros enteros al azar, el documento esta anexado con la clase Arreglo y tiene el titulo `millonDeNumeros.txt` y no requiere que se le pase ningún valor al constructor, solamente el nombre del archivo al método `leerArchivo(Nombre)`.

Algoritmo

1. Inicio.
2. Nueva clase “Arreglo” con los siguientes métodos:
 1. `void lecturaDatos()`
 2. `void imprimeArreglo()`
 3. `void heapSort()`
 4. `void cambio()`
 5. `void maxHeap`
 6. `crearHeap()`
 7. `radixSort()`
3. Y las variables
 1. `Integer[] arreglo`
 2. `Int nMaxDigitos`
 3. `Int tamaño`
 4. `Int izquierda`
 5. `Int derecha`
 6. `Int masGrande`

4. Funcionamiento:
 1. Arreglo objeto = new Arreglo()
 1. Se instancia el objeto.
 2. Void lecturaDeDatos(String nombre):
 1. Lee el archivo al cual se le pasa un nombre.
 2. Asigna los valores del archivo a []arreglo.
 3. Void imprimeArreglo():
 1. Imprime el []arreglo.
 4. void heapSort()
 1. Tamaño es igual al tamaño del arreglo - 1
 2. Para i que es igual a tamaño/2 i disminuye entonces se llama a maxHeap.
 3. Se llama a la función crearHeap pasándole el valor de arreglo.
 4. Hace un ciclo for que va desde el valor del temano del arreglo hasta 1.
 1. Llama al metodo cambio()
 2. Tamaño = tamaño - 1
 3. Llama al metodo maxHeap() y se le pasa el valor de arreglo y el valor de la constante del ciclo.
 5. void cambio(int i, int j)
 1. T = arreglo[i]
 2. arreglo[i] = arreglo[j]
 3. arreglo[j] = t
 6. Void maxHeap(int i)
 1. Izquierda = 2 * i
 2. Derecha = 2 * i + 1
 3. Si izquierda es menor o igual a tamaño y el arreglo en la posición izquierda es mayor que el valor del iterador i entonces:
 1. masGrande
 4. Sino entonces
 1. masGrande = i
 5. Si Derecha menor igual a tamaño y arreglo en la posición derecha es mayor que arreglo en la posición masGrande.
 6. Si mas grande es igual o diferente a i entonces se llama al método cambio y maxHeap.
 7. Void radixSort()
 1. Se crea un ArrayList de ArrayList de enteros llamado Bucket.
 2. Se crea un ArrayList de enteros llamado resultado.
 3. Se declara variable de tipo doble llamada temporal
 4. Se declara una variable de tipo doble llamada t
 5. Ciclo que va de uno al nMaxDigitos
 1. Temporal es igual a la potencia de 10 al numero de iteración menos uno.
 2. Ciclo que va de 0 a 9
 1. Se crea un nuevo ArrayList dentro del ArrayList.
 3. Ciclo que va de 0 al tamaño del arreglo.
 1. T = arreglo en j, entre temporal, modularidad 10.
 2. A bucket en el espacio t, se le asigna el arreglo en la posición j.
 4. Se asigna el valor de bucket a resultado.
 5. Arreglo es igual a resultado
 6. Fin.

Descripción Técnica

clase Arreglo

Declaracion de variables necesarias para el funcionamiento correcto del programa.

```

Integer[] arreglo
int nMaxDigitos
static int tamaño
static int iz
static int der
static int masGrande
void lecturaDatos(String archivo)

```

arreglo.

```

    try
        String line = null
        FileReader fileReader = new FileReader(archivo)
        BufferedReader bufferedReader = new BufferedReader(fileReader)
        int contador = 0

```

Se declara una instancia de FileReader para poder acceder a un archivo, asimismo una instancia de BufferedReader que obtendrá la información de este. Se declara un contador que nos servirá para identificar el primer valor del archivo, el cual será el tamaño del arreglo de tipo Integer.

```

        mientras((line = bufferedReader.readLine()) != null)
            si (contador == 0)
                arreglo = new Integer[Integer.parseInt(line)]
            sino si (contador > 0)
                arreglo[contador-1] = Integer.parseInt(line)
                si (nMaxDigitos < line.length())
                    nMaxDigitos = line.length()
            contador++

```

Mientras no sea el final del archivo, entonces se asignará el tamaño del arreglo y los valores que obtendrá este.

```

        bufferedReader.close()
        fileReader.close()
        Se cierran los lectores del archivo
        catch (FileNotFoundException ex)
            imprime("El archivo " + archivo + " no se encuentra en el directorio.")
        catch (IOException ex)
            Imprime("No se pudo leer el archivo " + archivo + ".")

```

Si no se encuentra el archivo o no se puede leer, entonces se muestra mensaje indicando la situación.

```

public void imprimeArreglo()
    try
        Imprime("Arreglo: " + Arrays.toString(arreglo))

        catch(NullPointerException arreglo)
            Imprime("El arreglo está vacío")

```

Se imprime el arreglo, si este no tiene elementos, se mandará un mensaje de error.

```

public void heapSort()
    //Se Crea el heap

```

```

tamano = arreglo.length - 1
por(int i = tamano/2 i >= 0 i--)
    maxHeap(i)
//Fin de creacion de heap

```

Se empiezan a acomodar los valores que se encuentran dentro del heap, utilizando las funciones cambio(int, int) y maxHeap(int).

```

por(int i = tamano i > 0 i--)
    cambio(0, i)
    tamano = tamano - 1
    maxHeap(0)

```

public void cambio(int i, int j) **Metodo Cambio para hacer cambio de lugar de los valores.**

```

int t = arreglo[i]
arreglo[i] = arreglo[j]
arreglo[j] = t

```

public void maxHeap(int i) **Metodo para reacomodar a cada hijo con su respectivo padre.**

```

iz = 2 * i
der = 2 * i + 1
si(iz <= tamano && arreglo[iz] > arreglo[i])

```

Si el valor del hijo izquierdo es menor o igual al tamaño y el arreglo en la posición del hijo izquierdo es mayor al arreglo en i, entonces iz será el valor mas grande, sino el masGrande será i.

```

    masGrande = iz
else
    masGrande = i

```

```

    si(der <= tamano && arreglo[der] > arreglo[masGrande])

```

Si el valor del hijo derecho es menor o igual al tamaño y el arreglo en la posición del hijo derecho es mayor al arreglo en masGrande, entonces der será el valor masGrande.

```

        masGrande = der

```

```

    si(masGrande != i)

```

Si mas grande es diferente a i se llamara a la funcion cambio, pasandole los valores de i y mas grande, cambiandolos de lugar y se llamara recursivamente a maxHeap pasandole el valor de masGrande.

```

        cambio(i, masGrande)
        maxHeap(masGrande)

```

```

public void radixSort()

```

Ordena los elemento del arreglo de la clase usando el algoritmo radixSort

```

    ArrayList<ArrayList<Integer>> bucket = new

```

```

    ArrayList<ArrayList<Integer>>()

```

ArrayList bidimensional para usar como cubeta

```
ArrayList<Integer> resultado = new ArrayList<Integer>()
```

ArrayList para asignar los valores de bucket en orden.

```
double temporal
```

```
int t
```

```
por (int i = 1 i <= nMaxDigitos i++)
```

Ciclo que va desde 1 hasta el numero de digitos mas grande para poder sacar el digito dependiendo de la iteracion i.

```
temporal = Math.pow(10.0, (double)i-1)
```

```
por (int j = 0 j <= 9 j++)
```

Ciclo que va desde 0 hasta el ultimo elemento del arreglo para crear nuevos ArrayList de tipo Integer en bucket, para tener una cubeta que valla del 0 al 9.

```
bucket.add(new ArrayList<Integer>())
```

```
por (int j = 0 j < arreglo.length j++)
```

Ciclo que va desde 0 al tamaño del arreglo, que saca el índice dependiendo de cual sea el dígito que se sacó, para después insertar en la cubeta el número completo que se encuentra en arreglo.

```
t = (int)((arreglo[j]/temporal)%10)
```

```
bucket.get(t).add(new Integer(arreglo[j]))
```

```
por (int j = 0 j <= 9 j++)
```

Ciclo que va desde 0 hasta el 9 para iterar entre los valores bidimensionales de bucket y poderlos asignar al ArrayList resultado.

```
por (int w = 0 w <= bucket.get(j).size() -1 w++)
```

```
resultado.add(bucket.get(j).get(w))
```

```
arreglo = resultado.toArray(arreglo)
```

Se asigna el valor de resultado al arreglo original y se limpian los valores de resultado y bucket.

```
resultado.clear()
```

```
bucket.clear()
```