Informe de Laboratorio 08

Tema: HashMap

Nota

Estudiante	Escuela	Asignatura
Julio Rubén Chura Acabana jchuraaca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	F. de Programción 2 Semestre: I Código: 20230472

Laboratorio	Tema	Duración
08	HashMap	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Octubre 2023	Al 23 Octubre 2023

1. Tarea

- Tendrá 2 Ejércitos. Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados (distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Usar HashMap.
- Deberá usar la búsqueda binaria y secuencial
- Usted debe realizar varios commits y al término de la actividad deberá realizar un informe.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- vim 9.0
- OpenJDK 64-Bits 20.0.2.

- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- HashMap

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JulioChura/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/JulioChura/fp2-23b/tree/main/fase01/lab08`

4. Actividades con el repositorio GitHub

4.1. Creación y modificación de métodos

Listing 1: Inicializando el espacio de trabajo

```
mkdir lab08
cd lab07
Copy-Item "Soldier.java" -Destination "..\lab08"
Copy-Item "VideoJuego4.java" -Destination "..\lab08\VideoJueg05.java"
cd ..
cd lab08
vim VideoJuego5.java
```

Listing 2: Commit: 04c6fccaf0efb49ff5850c22d6d70e304656daff

```
git add Soldier.java
git commit -m "Se copia la clase Soldier.java sin modificarla"
```

Listing 3: Se modifica el método que genera un HashMap cuya llave es un String y el elemento es un Soldier

```
vim VideoJuego5.java
```

```java
    public static HashMap<String, Soldier> generateArmy() {
        int rowBoard = 10;
        int columnBoard = 10;
        HashMap<String, Soldier> army = new HashMap<String, Soldier>();
        Random random = new Random();
        int amount = random.nextInt(10) + 1;
        int n = 0;
        do {
            int row = random.nextInt(rowBoard) + 1;
            int column = random.nextInt(columnBoard) + 1;
            String key = "Soldier" + row + "X" + column;
            if (!army.containsKey(key)) {
                int lifePoints = random.nextInt(5) + 1;
                Soldier soldier = new Soldier();
                soldier.setColumn(column);
                soldier.setRow(row);
                soldier.setLifePoints(lifePoints);
                soldier.setName(key);
                army.put(key, soldier);
                n++;
            }
        } while (n < amount);
        return army;
    }
```

- El método generateArmy() crea un HashMap cuya clave es de tipo String y es el nombre del Soldier. La razón de la elección de la clave es que las filas y columnas no se deben repetir por lo que guarda relación con la teoría que nos indica que las claves deben ser únicas.
- Para una mejor representación del Soldier en el tablero, se decide sumar uno a los números aleatorios que salgan de las row y column ya que el usuario no está acostumbrado a contar desde el 0.

Listing 4: Commit: 0c0b73fb81d0a4ced7ff7998e9ed8dc40ac99f17

```
git add VideoJuego5.java
git commit -m "Se corrigen cosas del metodo generateArmy"
git push -u origin main
```

Listing 5: Se modifica el método que genera un HashMap para el ejército B

```
vim VideoJuego5.java
```

```java
    public static HashMap<String, Soldier> generateArmyB(HashMap<String, Soldier> a) {
        int rowBoard = 10;
        int columnBoard = 10;
        HashMap<String, Soldier> army = new HashMap<String, Soldier>();
        Random random = new Random();
        int amount = random.nextInt(10) + 1;
        int n = 0;
        do {
            int row = random.nextInt(rowBoard) + 1;
            int column = random.nextInt(columnBoard) + 1;
            String key = "Soldier" + row + "X" + column;
            if (!army.containsKey(key) && !a.containsKey(key)) {
                int lifePoints = random.nextInt(5) + 1;
                Soldier soldier = new Soldier();
                soldier.setColumn(column);
                soldier.setRow(row);
                soldier.setLifePoints(lifePoints);
                soldier.setName(key);
                army.put(key, soldier);
                n++;
            }
        } while (n < amount);
        return army;
    }
```

- Este método recibe como parámetro un HashMap que es del ejercito A. Esto con el fin de garantizar que los elementos del ejército B se creén sin que hayan cruces con los del A.
- Se hace uso del método containsKey el cual nos retorna un boolean, siendo true que existe esa llave dentro del HashMap. Se hace uso en dos casos, uno es para garantizar que la llave que se ha creado sea única dentro del HashMap army y el otro para buscar coincidencias en el HashMap de a.

Listing 6: Commit: c78e6ca75a2f9f69cc4559c9a8b598596b86b947

```
git add VideoJuego5.java
git commit -m "Se crea el metodo generateArmyB"
git push -u origin main
```

Listing 7: Se modifica el método que imprime los datos de ambos ejércitos

```
vim VideoJuego5.java
```

```java
    public static void showByCreation(HashMap<String, Soldier> army) {
        for (String key : army.keySet()) {
            System.out.println(key + ": " + army.get(key));
        }
    }
```

- Se opta por modificar el método ya realizado de la actividad de laboratorio anterior por lo que ya se podrá imprimir HashMap.

Listing 8: Probando los métodos

```
javac VideoJuego5.java
java VideoJuego5
```

```
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier9X9: Soldier [name=Soldier9X9, lifePoints=3, row=10, column=10]
Soldier5X4: Soldier [name=Soldier5X4, lifePoints=1, row=6, column=5]
Soldier4X5: Soldier [name=Soldier4X5, lifePoints=2, row=5, column=6]
Soldier10X9: Soldier [name=Soldier10X9, lifePoints=3, row=11, column=10]
Soldier2X4: Soldier [name=Soldier2X4, lifePoints=4, row=3, column=5]
Soldier2X5: Soldier [name=Soldier2X5, lifePoints=1, row=3, column=6]
Soldier8X4: Soldier [name=Soldier8X4, lifePoints=1, row=9, column=5]
Soldier3X9: Soldier [name=Soldier3X9, lifePoints=3, row=4, column=10]
DATOS DEL EJRCITO B
Soldier5X3: Soldier [name=Soldier5X3, lifePoints=5, row=6, column=4]
Soldier2X7: Soldier [name=Soldier2X7, lifePoints=5, row=3, column=8]
Soldier2X8: Soldier [name=Soldier2X8, lifePoints=2, row=3, column=9]
Soldier2X10: Soldier [name=Soldier2X10, lifePoints=2, row=3, column=11]
Desea jugar una ronda?(si/no)
no
```

- No damos cuenta que la llave de cada Soldier no coincide con la fila y columna, este problema se debe a los accesores de row y column ya que a cada uno se les suma 1.

Listing 9: Commit: 8d7511cfd2f009ae04e1691cf9f2f85b3c80b0f7

```
git add VideoJuego5.java
git commit -m "Se modifica el metodo showByCreation"
git push -u origin main
```

Listing 10: Se modifica la clase Soldier

```
vim Soldier.java
```

```
32      public void setRow(int row) {
33  -       this.row = row + 1;
34      }
35
36      public void setColumn(int column) {
37  -       this.column = column + 1;
38      }
39
```

```
32      public void setRow(int row) {
33  +       this.row = row;
34      }
35
36      public void setColumn(int column) {
37  +       this.column = column;
38      }
39
```

- Se le borra +1 tanto a los métodos setColumn y setRow ya que de no hacerlo se presentaría el problema de desbordamiento.

Listing 11: Compilando y probando

```
javac VideoJuego5.java
java VideoJuego5
```

```
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier3X2: Soldier [name=Soldier3X2, lifePoints=4, row=3, column=2]
Soldier6X1: Soldier [name=Soldier6X1, lifePoints=2, row=6, column=1]
Soldier10X7: Soldier [name=Soldier10X7, lifePoints=3, row=10, column=7]
Soldier8X8: Soldier [name=Soldier8X8, lifePoints=1, row=8, column=8]
DATOS DEL EJRCITO B
Soldier7X1: Soldier [name=Soldier7X1, lifePoints=5, row=7, column=1]
Soldier10X9: Soldier [name=Soldier10X9, lifePoints=3, row=10, column=9]
Soldier5X1: Soldier [name=Soldier5X1, lifePoints=1, row=5, column=1]
Soldier10X5: Soldier [name=Soldier10X5, lifePoints=5, row=10, column=5]
Desea jugar una ronda?(si/no)
no
```

- Ahora si coinciden las llaves y la posición de cada Soldier.

Listing 12: Commit: a13e2f09357ac8e78d9aa0a0c1aa9e654a0a09df

```
git add VideoJuego5.java
git commit -m "Se borra mas uno en los setColumn y setRow"
git push -u origin main
```

Listing 13: Se modifica el metodo LongerLife

```
vim VideoJuego5.java
```

```
1    InsertionSort(arr[], n)
2        for i = 1 to n-1
3        key = arr[i]
4        j = i-1
5        while j >= 0 and arr[j] > key
6            arr[j+1] = arr[j]
7            j = j-1
8        arr[j+1] = key
```

- Este el pseudocódigo del ordenamiento por inserción a un Arreglo Estándar.

```java
    public static Soldier longerLife(HashMap<String, Soldier> hashMap) {
        Soldier[] army = new Soldier[hashMap.size()];
        int index = 0;
        for (String key : hashMap.keySet()) {
            army[index] = hashMap.get(key);
            index++;
        }
        int n = army.length;
        for (int i = 1; i < n; i++) {
            Soldier key = army[i];
            int j = i - 1;
            while (j >= 0 && army[j].getLifePoints() > key.getLifePoints()) {
                army[j + 1] = army[j];
                j--;
            }
            army[j + 1] = key;
        }
        return army[army.length - 1];
    }
```

- Recordemos que un Hash Map no se puede ordenar de manera directa, por lo que se opta de pasar los elementos del Hash Map a un arreglo Estándar de tipo Soldier. No nos preocupamos por las llaves ya que prácticamente vendría a ser lo mismo que los nombres de cada Soldier.
- Una vez trasladados los elementos al arreglo Estándar, se hace uso del algoritmo de ordenamiento por inserción. Finalmente se retorna un Soldier de la última posición del arreglo porque está ordenado de manera ascendente.

Listing 14: Compilando y probando

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier4X4: Soldier [name=Soldier4X4, lifePoints=2, row=4, column=4]
Soldier5X2: Soldier [name=Soldier5X2, lifePoints=2, row=5, column=2]
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=1, row=9, column=3]
Soldier4X8: Soldier [name=Soldier4X8, lifePoints=2, row=4, column=8]
Soldier4X7: Soldier [name=Soldier4X7, lifePoints=3, row=4, column=7]
Soldier9X2: Soldier [name=Soldier9X2, lifePoints=2, row=9, column=2]
Soldier6X8: Soldier [name=Soldier6X8, lifePoints=5, row=6, column=8]
Mayor vida en A: Soldier [name=Soldier6X8, lifePoints=5, row=6, column=8]
DATOS DEL EJRCITO B
Soldier4X9: Soldier [name=Soldier4X9, lifePoints=4, row=4, column=9]
Soldier3X7: Soldier [name=Soldier3X7, lifePoints=3, row=3, column=7]
Soldier9X7: Soldier [name=Soldier9X7, lifePoints=5, row=9, column=7]
Mayor vida en B: Soldier [name=Soldier9X7, lifePoints=5, row=9, column=7]
Desea jugar una ronda?(si/no)
no
```

Listing 15: Commit: 34a06097f486f0e96e4785f39e9311c655287aa7

```
git add VideoJuego5.java
git commit -m "Se adapata el metodo longerLife para un HashMap"
git push -u origin main
```

Listing 16: Se modifica el método retorna la suma de la cantidad de puntos del ejército

```
vim VideoJuego5.java
```

```
141     public static int totalLife(HashMap<String, Soldier> army) {
142         int addition = 0;
143         for (String key : army.keySet()) {
144             addition = addition + army.get(key).getLifePoints();
145         }
146         return addition;
```

- Se hace uso del método keySet para poder acceder a todas llaves y valores. Luego de sacar el valor se le saca sus puntos de vida con el método getLifePoints y se procede a ir sumando los puntos de vida.

Listing 17: Compilando y probando

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
ooooooooooooooo FASE 1 DE LA CONTIENDA ooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier1X10: Soldier [name=Soldier1X10, lifePoints=2, row=1, column=10]
Soldier5X2: Soldier [name=Soldier5X2, lifePoints=2, row=5, column=2]
Soldier7X5: Soldier [name=Soldier7X5, lifePoints=1, row=7, column=5]
Soldier8X4: Soldier [name=Soldier8X4, lifePoints=1, row=8, column=4]
Soldier5X8: Soldier [name=Soldier5X8, lifePoints=4, row=5, column=8]
Soldier8X8: Soldier [name=Soldier8X8, lifePoints=3, row=8, column=8]
Soldier8X9: Soldier [name=Soldier8X9, lifePoints=1, row=8, column=9]
Soldier3X10: Soldier [name=Soldier3X10, lifePoints=5, row=3, column=10]
Mayor vida en A: Soldier [name=Soldier3X10, lifePoints=5, row=3, column=10]
El total de vida del ejercito A es: 19
DATOS DEL EJRCITO B
Soldier9X9: Soldier [name=Soldier9X9, lifePoints=5, row=9, column=9]
Soldier2X1: Soldier [name=Soldier2X1, lifePoints=2, row=2, column=1]
Soldier3X5: Soldier [name=Soldier3X5, lifePoints=2, row=3, column=5]
Soldier4X2: Soldier [name=Soldier4X2, lifePoints=5, row=4, column=2]
Soldier9X8: Soldier [name=Soldier9X8, lifePoints=5, row=9, column=8]
Mayor vida en B: Soldier [name=Soldier9X8, lifePoints=5, row=9, column=8]
El total de vida del ejercito B es: 19
Desea jugar una ronda?(si/no)
no
```

Listing 18: Commit: 8c68956fc2e3f75c1fd1b3c7bd5919f5b50c9416

```
git add VideoJuego5.java
git commit -m "Se culmina el metodo totalLife"
git push -u origin main
```

Listing 19: Se calcula el promedio de vida de cada ejército

```
vim VideoJuego5.java
```

```
14            int a = totalLife(armyA);
15            int b = totalLife(armyB);
16
17            System.out.println("ooooooooooooooo  FASE 1 DE LA CONTIENDA  ooooooooooooooo");
18            System.out.println("Mostrando estadisticas de cada ejercito" + "\n");
19            System.out.println("Mostrando soldados por orden de creacion");
20            System.out.println("DATOS DEL DEL EJERCITO A");
21            showByCreation(armyA);
22            System.out.println("Mayor vida en A: " + longerLife(armyA));
23            System.out.println("El total de vida del ejercito A es: " + a);
24            System.out.println("El promedio de vida del ejercito A es: " + (double) a / armyA.size());
25            System.out.println("DATOS DEL EJRCITO B");
26            showByCreation(armyB);
27            System.out.println("Mayor vida en B: " + longerLife(armyB));
28            System.out.println("El total de vida del ejercito B es: " + b);
29            System.out.println("El promedio de vida del ejercito B es: " + (double) b / armyB.size());
```

- En la línea 14 y 15 de la imagen, se puede observar que los puntos de vida total de cada ejército son recibidos por dos variables.
- En la línea 24 se observa que se imprime un mensaje que nos indica el promedio de vida del ejército A. Del mismo modo, en la linea 29 se hace lo mismo, pero para el ejército B.

Listing 20: Commit: ac8683975133d422bba4e6f0c13a5c378a38e50d

```
git add VideoJuego5.java
git commit -m "Se calcula el promedio de vida de cada ejercito"
git push -u origin main
```

Listing 21: Se modifica el método que imprime el ranking de poder

```
vim VideoJuego5.java
```

```
149   public static void orderByPower(HashMap<String, Soldier> army) {
150       ArrayList<Map.Entry<String, Soldier>> list = new ArrayList<>(army.entrySet());
151       Collections.sort(list, new Comparator<Map.Entry<String, Soldier>>() {
152           public int compare(Map.Entry<String, Soldier> o1, Map.Entry<String, Soldier> o2) {
153               return Integer.compare(o1.getValue().getLifePoints(), o2.getValue().getLifePoints());
154           }
155       });
156       HashMap<String, Soldier> sortedMap = new LinkedHashMap<>();
157       for (Map.Entry<String, Soldier> entry : list) {
158           sortedMap.put(entry.getKey(), entry.getValue());
159       }
160       for (String key : sortedMap.keySet()) {
161           System.out.println(key + ": " + sortedMap.get(key));
162       }
163
164   }
```

- Como ya se mencionó, un HashMap no puede ser ordenado de forma directa. Más antes se propuso un enfoque de trasladar los datos a un arreglo Estándar y luego aplicar el algoritmo de inserción, no obstante, en este método se propone usar un método de los List el cual es Collections.sort, sin embargo, no puede ser aplicado a un Map, por lo que primero se opta por pasar los datos a un ArrayList y luego modificar el Comparator para personalizar las comparaciones que determinarán el orden. Luego movemos esos resultados a un LinkedHashMap el cual mantiene el orden. Finalmente se mostrarán los elementos ya ordenados.

Listing 22: Compilando y probando

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
ooooooooooooooo FASE 1 DE LA CONTIENDA ooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier5X3: Soldier [name=Soldier5X3, lifePoints=1, row=5, column=3]
Soldier6X3: Soldier [name=Soldier6X3, lifePoints=3, row=6, column=3]
Soldier10X8: Soldier [name=Soldier10X8, lifePoints=4, row=10, column=8]
Soldier2X4: Soldier [name=Soldier2X4, lifePoints=1, row=2, column=4]
Soldier8X2: Soldier [name=Soldier8X2, lifePoints=5, row=8, column=2]
Mayor vida en A: Soldier [name=Soldier8X2, lifePoints=5, row=8, column=2]
El total de vida del ejercito A es: 14
El promedio de vida del ejercito A es: 2.8
Mostrando soldados por ranking de poder de A
Soldier5X3: Soldier [name=Soldier5X3, lifePoints=1, row=5, column=3]
Soldier2X4: Soldier [name=Soldier2X4, lifePoints=1, row=2, column=4]
Soldier6X3: Soldier [name=Soldier6X3, lifePoints=3, row=6, column=3]
Soldier10X8: Soldier [name=Soldier10X8, lifePoints=4, row=10, column=8]
Soldier8X2: Soldier [name=Soldier8X2, lifePoints=5, row=8, column=2]
DATOS DEL EJRCITO B
Soldier2X5: Soldier [name=Soldier2X5, lifePoints=4, row=2, column=5]
Soldier5X8: Soldier [name=Soldier5X8, lifePoints=1, row=5, column=8]
Soldier7X3: Soldier [name=Soldier7X3, lifePoints=1, row=7, column=3]
Soldier5X9: Soldier [name=Soldier5X9, lifePoints=4, row=5, column=9]
Mayor vida en B: Soldier [name=Soldier5X9, lifePoints=4, row=5, column=9]
El total de vida del ejercito B es: 10
El promedio de vida del ejercito B es: 2.5
Mostrando soldados por ranking de poder de B
Soldier5X8: Soldier [name=Soldier5X8, lifePoints=1, row=5, column=8]
Soldier7X3: Soldier [name=Soldier7X3, lifePoints=1, row=7, column=3]
Soldier2X5: Soldier [name=Soldier2X5, lifePoints=4, row=2, column=5]
Soldier5X9: Soldier [name=Soldier5X9, lifePoints=4, row=5, column=9]
Desea jugar una ronda?(si/no)
no
```

Listing 23: Commit: d3af64e3678b36984d744ac19d598ea08e012bee

```
git add VideoJuego5.java
git commit -m "Metodo orderByPower culminado"
git push -u origin main
```

Listing 24: Se implementa el método que hace una búsqueda binaria del nombre solicitado

```
vim VideoJuego5.java
```

```
function busqueda_binaria (datos, valor_busqueda):
    inicio = 0
    fin = longitud (datos) - 1
    mientras inicio <= fin:
        medio = (inicio + fin) // 2
        si datos [medio] == valor_busqueda:
            devolver medio
        sino si datos [medio] < valor_busqueda:
            inicio = medio + 1
        sino:
            fin = medio - 1
    devolver -1
```

- Este es el pseudocódigo de búsqueda binaria.

```
163     public static void binarySearchByName(HashMap<String, Soldier> army, String name) {
164         ArrayList<Map.Entry<String, Soldier>> list = new ArrayList<>(army.entrySet());
165         Collections.sort(list, new Comparator<Map.Entry<String, Soldier>>() {
166             public int compare(Map.Entry<String, Soldier> o1, Map.Entry<String, Soldier> o2) {
167                 return o1.getValue().getName().compareTo(o2.getValue().getName());
168             }
169         });
170         HashMap<String, Soldier> sortedMap = new LinkedHashMap<>();
171         for (Map.Entry<String, Soldier> entry : list) {
172             sortedMap.put(entry.getKey(), entry.getValue());
173         }
174         ArrayList<String> keys = new ArrayList<>(sortedMap.keySet());
175         int low = 0;
176         int high = keys.size() - 1;
177         while (low <= high) {
178             int mid = (low + high) / 2;
179             String str = keys.get(mid);
180             if (str.equalsIgnoreCase(name)) {
181                 System.out.println("Se ha encontrado: " + army.get(str));
182                 return;
183             } else if (str.compareTo(name) < 0) {
184                 low = mid + 1;
185             } else {
186                 high = mid - 1;
187             }
188         }
189         System.out.println("No se han encontrado coincidencias");
190 
191     }
```

- Primero, se convierten los elementos del HashMap en una lista de Map.Entry, que contiene tanto las claves (nombres) como los valores (objetos Soldier). Luego se ordena esta lista de Map.Entry en función del nombre de los Soldier utilizando Collections.sort. Posteriormente se crea un nuevo LinkedHashMap llamado sortedMap para mantener los elementos ordenados, y se copian los elementos ordenados en él. Luego se crea un ArrayList llamado keys que contiene las llaves en

el orden deseado, que es el orden alfabético de los nombres de los soldados. Luego se realiza la búsqueda binaria en el ArrayList de llaves (keys).

Listing 25: Compilando y probando

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier9X4: Soldier [name=Soldier9X4, lifePoints=1, row=9, column=4]
Soldier5X5: Soldier [name=Soldier5X5, lifePoints=1, row=5, column=5]
Mayor vida en A: Soldier [name=Soldier5X5, lifePoints=1, row=5, column=5]
El total de vida del ejercito A es: 2
El promedio de vida del ejercito A es: 1.0
Mostrando soldados por ranking de poder de A
Soldier9X4: Soldier [name=Soldier9X4, lifePoints=1, row=9, column=4]
Soldier5X5: Soldier [name=Soldier5X5, lifePoints=1, row=5, column=5]
Ingrese el nombre del Soldier que desea buscar
soldier9x4
Se ha encontrado: Soldier [name=Soldier9X4, lifePoints=1, row=9, column=4]
DATOS DEL EJRCITO B
Soldier4X8: Soldier [name=Soldier4X8, lifePoints=1, row=4, column=8]
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=2, row=9, column=3]
Soldier10X1: Soldier [name=Soldier10X1, lifePoints=2, row=10, column=1]
Mayor vida en B: Soldier [name=Soldier10X1, lifePoints=2, row=10, column=1]
El total de vida del ejercito B es: 5
El promedio de vida del ejercito B es: 1.6666666666666667
Mostrando soldados por ranking de poder de B
Soldier4X8: Soldier [name=Soldier4X8, lifePoints=1, row=4, column=8]
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=2, row=9, column=3]
Soldier10X1: Soldier [name=Soldier10X1, lifePoints=2, row=10, column=1]
Ingrese el nombre del Soldier que desea buscar
soldier9x3
Se ha encontrado: Soldier [name=Soldier9X3, lifePoints=2, row=9, column=3]
Desea jugar una ronda?(si/no)
no
```

Listing 26: Commit: 556b06b12e835c82e309c4ad3dcde5ced22e2b94

```
git add VideoJuego5.java
git commit -m "Metodo binarySearchByName culminado"
git push -u origin main
```

Listing 27: Se implementa el método que hace la búsqueda secuencial de un Soldier por su nombre

```
vim VideoJuego5.java
```

```
193     public static void sequenceSearchByName(HashMap<String, Soldier> army, String name) {
194         ArrayList<String> keys = new ArrayList<>(army.keySet());
195         for (int i = 0; i < army.size(); i++) {
196             String myKey = keys.get(i);
197             if (name.equalsIgnoreCase(myKey)) {
198                 System.out.println(army.get(myKey));
199             }
200         }
201     }
```

- Debido a condiciones de la práctica de laboratorio, se añade un método que hace una búsqueda secuencial. Para lograr ello, se decide guardar las llaves en un ArrayList con la ayuda del método keySet() que está dentro del constructor de ArrayList. Una vez realizado ello, ya podremos hacer uso de la búsqueda secuencial donde se compara la llave (recordemos que la llave es un dato de tipo String) con el String name del parámetro, cuando hayan coincidencias, el valor actual de la variable myKey será usada para poder acceder al valor del HashMap.

Listing 28: Compilando y probando

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier4X4: Soldier [name=Soldier4X4, lifePoints=1, row=4, column=4]
Soldier10X4: Soldier [name=Soldier10X4, lifePoints=5, row=10, column=4]
Soldier2X8: Soldier [name=Soldier2X8, lifePoints=1, row=2, column=8]
Soldier4X6: Soldier [name=Soldier4X6, lifePoints=3, row=4, column=6]
Mayor vida en A: Soldier [name=Soldier10X4, lifePoints=5, row=10, column=4]
El total de vida del ejercito A es: 10
El promedio de vida del ejercito A es: 2.5
Mostrando soldados por ranking de poder de A
Soldier4X4: Soldier [name=Soldier4X4, lifePoints=1, row=4, column=4]
Soldier2X8: Soldier [name=Soldier2X8, lifePoints=1, row=2, column=8]
Soldier4X6: Soldier [name=Soldier4X6, lifePoints=3, row=4, column=6]
Soldier10X4: Soldier [name=Soldier10X4, lifePoints=5, row=10, column=4]
Ingrese el nombre del Soldier que desea buscar
soldier4x6
Se ha encontrado: Soldier [name=Soldier4X6, lifePoints=3, row=4, column=6]
DATOS DEL EJRCITO B
Soldier3X1: Soldier [name=Soldier3X1, lifePoints=5, row=3, column=1]
Soldier2X2: Soldier [name=Soldier2X2, lifePoints=2, row=2, column=2]
Soldier4X1: Soldier [name=Soldier4X1, lifePoints=3, row=4, column=1]
Soldier2X6: Soldier [name=Soldier2X6, lifePoints=5, row=2, column=6]
Soldier1X7: Soldier [name=Soldier1X7, lifePoints=2, row=1, column=7]
Soldier7X2: Soldier [name=Soldier7X2, lifePoints=5, row=7, column=2]
Soldier4X3: Soldier [name=Soldier4X3, lifePoints=4, row=4, column=3]
Soldier10X5: Soldier [name=Soldier10X5, lifePoints=1, row=10, column=5]
Mayor vida en B: Soldier [name=Soldier7X2, lifePoints=5, row=7, column=2]
El total de vida del ejercito B es: 27
El promedio de vida del ejercito B es: 3.375
Mostrando soldados por ranking de poder de B
Soldier10X5: Soldier [name=Soldier10X5, lifePoints=1, row=10, column=5]
```

```
Soldier2X2: Soldier [name=Soldier2X2, lifePoints=2, row=2, column=2]
Soldier1X7: Soldier [name=Soldier1X7, lifePoints=2, row=1, column=7]
Soldier4X1: Soldier [name=Soldier4X1, lifePoints=3, row=4, column=1]
Soldier4X3: Soldier [name=Soldier4X3, lifePoints=4, row=4, column=3]
Soldier3X1: Soldier [name=Soldier3X1, lifePoints=5, row=3, column=1]
Soldier2X6: Soldier [name=Soldier2X6, lifePoints=5, row=2, column=6]
Soldier7X2: Soldier [name=Soldier7X2, lifePoints=5, row=7, column=2]
Ingrese el nombre del Soldier que desea buscar
soldier1x7
Soldier [name=Soldier1X7, lifePoints=2, row=1, column=7]
Desea jugar una ronda?(si/no)
no
```

Listing 29: Commit: a0365cf1eebf124b7cfd8019ec7895186cb90074

```
git add VideoJuego5.java
git commit -m "Se culmina el metodo de busqueda sencuencial"
git push -u origin main
```

Listing 30: Se modifica el método que imprime el tablero

```
vim VideoJuego5.java
```

```
202     public static void myBoard(HashMap<String, Soldier> a, HashMap<String, Soldier> b) {
203         String[][] tablero = new String[10][10];
204         for (int i = 0; i < tablero.length; i++) {
205             for (int j = 0; j < tablero[i].length; j++) {
206                 tablero[i][j] = "|____|";
207             }
208         }
209         for (String key : a.keySet()) {
210             Soldier soldier = a.get(key);
211             int lifePoints = soldier.getLifePoints();
212             int row = soldier.getRow() - 1;
213             int column = soldier.getColumn() - 1;
214             String str = "|_" + "a" + lifePoints + "_|";
215             tablero[row][column] = str;
216         }
217         for (String key : b.keySet()) {
218             Soldier soldier = b.get(key);
219             int lifePoints = soldier.getLifePoints();
220             int row = soldier.getRow() - 1;
221             int column = soldier.getColumn() - 1;
222             String str = "|_" + "b" + lifePoints + "_|";
223             tablero[row][column] = str;
224         }
225         System.out.print("   A     B     C     D     E    F     G     H     I     J \n");
226         for (int i = 0; i < tablero.length; i++) {
227             System.out.printf("%2d", (i + 1));
228             for (int j = 0; j < tablero[i].length; j++) {
229                 System.out.print(tablero[i][j]);
230             }
231             System.out.println();
232         }
233     }
```

- Se hace uso de 4 ciclos for. El primero genera el tablero. El segundo ubica las fichas del ejército A. El tercer for acomoda las fichas del ejército B. El cuarto for que sería anidado muestra el tablero con las fichas incluidas.

Listing 31: Compilando y probando el codigo

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier7X3: Soldier [name=Soldier7X3, lifePoints=2, row=7, column=3]
Soldier4X6: Soldier [name=Soldier4X6, lifePoints=2, row=4, column=6]
Soldier8X3: Soldier [name=Soldier8X3, lifePoints=3, row=8, column=3]
Mayor vida en A: Soldier [name=Soldier8X3, lifePoints=3, row=8, column=3]
El total de vida del ejercito A es: 7
El promedio de vida del ejercito A es: 2.3333333333333335
Mostrando soldados por ranking de poder de A
Soldier7X3: Soldier [name=Soldier7X3, lifePoints=2, row=7, column=3]
Soldier4X6: Soldier [name=Soldier4X6, lifePoints=2, row=4, column=6]
Soldier8X3: Soldier [name=Soldier8X3, lifePoints=3, row=8, column=3]
Ingrese el nombre del Soldier que desea buscar
soldier5x3
No se han encontrado coincidencias
DATOS DEL EJRCITO B
Soldier3X1: Soldier [name=Soldier3X1, lifePoints=4, row=3, column=1]
Soldier6X2: Soldier [name=Soldier6X2, lifePoints=5, row=6, column=2]
Soldier8X1: Soldier [name=Soldier8X1, lifePoints=3, row=8, column=1]
Soldier1X6: Soldier [name=Soldier1X6, lifePoints=1, row=1, column=6]
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=4, row=9, column=3]
Soldier6X6: Soldier [name=Soldier6X6, lifePoints=1, row=6, column=6]
Soldier7X6: Soldier [name=Soldier7X6, lifePoints=3, row=7, column=6]
Mayor vida en B: Soldier [name=Soldier6X2, lifePoints=5, row=6, column=2]
El total de vida del ejercito B es: 21
El promedio de vida del ejercito B es: 3.0
Mostrando soldados por ranking de poder de B
Soldier1X6: Soldier [name=Soldier1X6, lifePoints=1, row=1, column=6]
Soldier6X6: Soldier [name=Soldier6X6, lifePoints=1, row=6, column=6]
Soldier8X1: Soldier [name=Soldier8X1, lifePoints=3, row=8, column=1]
Soldier7X6: Soldier [name=Soldier7X6, lifePoints=3, row=7, column=6]
Soldier3X1: Soldier [name=Soldier3X1, lifePoints=4, row=3, column=1]
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=4, row=9, column=3]
Soldier6X2: Soldier [name=Soldier6X2, lifePoints=5, row=6, column=2]
Ingrese el nombre del Soldier que desea buscar
soldier3x1
Soldier [name=Soldier3X1, lifePoints=4, row=3, column=1]

oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B     C     D     E     F     G     H     I     J
1 |____||____||____||____||____||_b1_||____||____||____||____|
2 |____||____||____||____||____||____||____||____||____||____|
```

```
3 |_b4_||____||____||____||____||____||____||____||____||____|
4 |____||____||____||____||____||_a2_||____||____||____||____|
5 |____||____||____||____||____||____||____||____||____||____|
6 |____||_b5_||____||____||____||_b1_||____||____||____||____|
7 |____||____||_a2_||____||____||_b3_||____||____||____||____|
8 |_b3_||____||_a3_||____||____||____||____||____||____||____|
9 |____||____||_b4_||____||____||____||____||____||____||____|
10|____||____||____||____||____||____||____||____||____||____|

Desea jugar una ronda?(si/no)
no
```

Listing 32: Commit: 6a85e033318ba91b893414da2a57263477c07c71

```
git add VideoJuego5.java
git commit -m "Metodo myBoard ha sido terminado"
git push -u origin main
```

4.2. Métodos que ya estaban y que no fueron modificados

```
233     public static void theWinner(int a, int b) {
234         if (a < b) {
235             System.out.println("El ganador es el equipo B");
236             System.out.println("Ventaja de " + (b - a) + " puntos de vida");
237         } else if (a > b) {
238             System.out.println("El ganador es el equipo A");
239             System.out.println("Ventaja de " + (a - b) + " puntos de vida");
240         } else {
241             System.out.println("Fue un empate");
242         }
243     }
```

- theWinner: Este método trabajado en la actividad pasada determina al ganador o si hubo empate.

```
245     public static boolean validation() {
246         Scanner sc = new Scanner(System.in);
247         do {
248             System.out.println("Desea jugar una ronda?(si/no)");
249             String answer = sc.next();
250             if (answer.equalsIgnoreCase("Si")) {
251                 return true;
252             } else if (answer.equalsIgnoreCase("No")) {
253                 return false;
254             } else {
255                 System.out.println("Respuesta no admsible");
256             }
257         } while (true);
258 
259     }
```

- validation: Este método hace psoible que el programa sea iterativo.

Listing 33: Compilando y probando el codigo en su versión final

```
javac VideoJuego5.java
java VideoJuego5
Desea jugar una ronda?(si/no)
si
oooooooooooooooo FASE 1 DE LA CONTIENDA oooooooooooooooo
Mostrando estadisticas de cada ejercito

Mostrando soldados por orden de creacion
DATOS DEL DEL EJERCITO A
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=4, row=9, column=3]
Soldier4X9: Soldier [name=Soldier4X9, lifePoints=4, row=4, column=9]
Soldier2X9: Soldier [name=Soldier2X9, lifePoints=5, row=2, column=9]
Soldier4X7: Soldier [name=Soldier4X7, lifePoints=5, row=4, column=7]
Soldier7X7: Soldier [name=Soldier7X7, lifePoints=1, row=7, column=7]
Soldier8X7: Soldier [name=Soldier8X7, lifePoints=2, row=8, column=7]
Mayor vida en A: Soldier [name=Soldier4X7, lifePoints=5, row=4, column=7]
El total de vida del ejercito A es: 21
El promedio de vida del ejercito A es: 3.5
Mostrando soldados por ranking de poder de A
Soldier7X7: Soldier [name=Soldier7X7, lifePoints=1, row=7, column=7]
Soldier8X7: Soldier [name=Soldier8X7, lifePoints=2, row=8, column=7]
Soldier9X3: Soldier [name=Soldier9X3, lifePoints=4, row=9, column=3]
Soldier4X9: Soldier [name=Soldier4X9, lifePoints=4, row=4, column=9]
Soldier2X9: Soldier [name=Soldier2X9, lifePoints=5, row=2, column=9]
Soldier4X7: Soldier [name=Soldier4X7, lifePoints=5, row=4, column=7]
Ingrese el nombre del Soldier que desea buscar
soldier9x3
Se ha encontrado: Soldier [name=Soldier9X3, lifePoints=4, row=9, column=3]
DATOS DEL EJRCITO B
Soldier8X3: Soldier [name=Soldier8X3, lifePoints=2, row=8, column=3]
Soldier9X8: Soldier [name=Soldier9X8, lifePoints=5, row=9, column=8]
Soldier2X10: Soldier [name=Soldier2X10, lifePoints=5, row=2, column=10]
Mayor vida en B: Soldier [name=Soldier2X10, lifePoints=5, row=2, column=10]
El total de vida del ejercito B es: 12
El promedio de vida del ejercito B es: 4.0
Mostrando soldados por ranking de poder de B
Soldier8X3: Soldier [name=Soldier8X3, lifePoints=2, row=8, column=3]
Soldier9X8: Soldier [name=Soldier9X8, lifePoints=5, row=9, column=8]
Soldier2X10: Soldier [name=Soldier2X10, lifePoints=5, row=2, column=10]
Ingrese el nombre del Soldier que desea buscar
soldier8x3
Soldier [name=Soldier8X3, lifePoints=2, row=8, column=3]

oooooooooooooooo FASE 2 DE LA CONTIENDA oooooooooooooooo
Mostrando el tablero de juego
   A     B     C     D     E     F     G     H     I     J
1 |____||____||____||____||____||____||____||____||____||____|
2 |____||____||____||____||____||____||____||____||_a5_||_b5_|
3 |____||____||____||____||____||____||____||____||____||____|
4 |____||____||____||____||____||____||_a5_||____||_a4_||____|
5 |____||____||____||____||____||____||____||____||____||____|
6 |____||____||____||____||____||____||____||____||____||____|
```

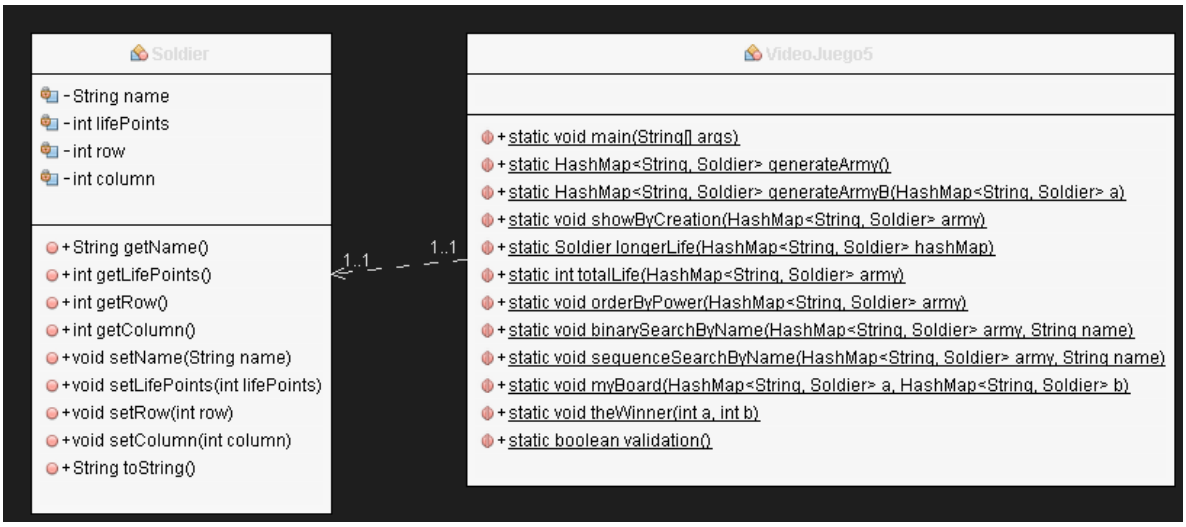
```
7 |____||____||____||____||____||____||_a1_||____||____||____|
8 |____||____||_b2_||____||____||____||_a2_||____||____||____|
9 |____||____||_a4_||____||____||____||____||_b5_||____||____|
10|____||____||____||____||____||____||____||____||____||____|

+++++++++++++++++ FASE 3 DE LA CONTIENDA +++++++++++++++++
El ganador se determina en base a los puntos de vida total
Enfrentamiento
El ganador es el equipo A
Ventaja de 9 puntos de vida
Desea jugar una ronda?(si/no)
no
```

Listing 34: Commit: 4343ad638f404dafc97349ff35c8808998ea0f1b

```
git add VideoJuego5.java
git commit -m "En el main se prepara lo necesario para determinar al ganador"
git push -u origin main
```

4.3. Diagrama UML y el main



```java
public class VideoJuego5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (validation()) {
            HashMap<String, Soldier> armyA = generateArmy();
            HashMap<String, Soldier> armyB = generateArmyB(armyA);
            int a = totalLife(armyA);
            int b = totalLife(armyB);

            System.out.println("oooooooooooooooo  FASE 1 DE LA CONTIENDA  oooooooooooooooo");
            System.out.println("Mostrando estadisticas de cada ejercito" + "\n");
            System.out.println("Mostrando soldados por orden de creacion");
            System.out.println("DATOS DEL DEL EJERCITO A");
            showByCreation(armyA);
            System.out.println("Mayor vida en A: " + longerLife(armyA));
            System.out.println("El total de vida del ejercito A es: " + a);
            System.out.println("El promedio de vida del ejercito A es: " + (double) a / armyA.size());
            System.out.println("Mostrando soldados por ranking de poder de A");
            orderByPower(armyA);
            System.out.println("Ingrese el nombre del Soldier que desea buscar");
            String nameA = sc.next();
            binarySearchByName(armyA, nameA);
            System.out.println("DATOS DEL EJRCITO B");
            showByCreation(armyB);
            System.out.println("Mayor vida en B: " + longerLife(armyB));
            System.out.println("El total de vida del ejercito B es: " + b);
            System.out.println("El promedio de vida del ejercito B es: " + (double) b / armyB.size());
            System.out.println("Mostrando soldados por ranking de poder de B");
            orderByPower(armyB);
            System.out.println("Ingrese el nombre del Soldier que desea buscar");
            String nameB = sc.next();
            sequenceSearchByName(armyB, nameB);
            System.out.println();

            System.out.println("oooooooooooooooo  FASE 2 DE LA CONTIENDA  oooooooooooooooo");
            System.out.println("Mostrando el tablero de juego");
            myBoard(armyA, armyB);
            System.out.println();

            System.out.println("++++++++++++++++++   FASE 3 DE LA CONTIENDA  ++++++++++++++++++");
            System.out.println("El ganador se determina en base a los puntos de vida total");
            System.out.println("Enfrentamiento");
            theWinner(a, b);

        }

    }
```

4.4. Estructura de laboratorio 08

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab08
|
|   Soldier.java
|   VideoJuego5.java
|
 ----latex
     |   programacion_lab08_rescobedoq_v1.0.pdf
     |   programacion_lab08_rescobedoq_v1.0.tex
     |
      ----img
              average.jpg
              binary.jpg
              binarySearchByName.jpg
              burbuja.jpg
              classSoldier.jpg
              generateArmy.jpg
```

```
generateArmyB.jpg
insertion.jpg
logo_abet.png
logo_episunsa.png
logo_unsa.jpg
longerLife.jpg
main.jpg
myBoard.jpg
orderByPower.jpg
sequence.jpg
showByCreation.jpg
theWinner.jpg
totalLife.jpg
uml.png
validation.jpg
```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
Total		20		18	

6. Referencias

- https://www.geeksforgeeks.org/binary-search/
- https://www.geeksforgeeks.org/insertion-sort/