

## Informe de Laboratorio 20

**Tema: Clase Ejército – Soldado – Mapa. Herencia y Polimorfismo. Miembros de clase**

Nota

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

Laboratorio	Tema	Duración
20	Clase Ejército – Soldado – Mapa. Herencia y Polimorfismo. Miembros de clase	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 8 de Enero 2023	Al 10 de Enero 2023

### 1. Tarea

- En este laboratorio deberá hacer uso de los conceptos de Herencia, Polimorfismo y composición. Para ello deberá crear diversas clases y culminada la actividad, deberá elaborar un informe

### 2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- Visual Studio Code 1.85.1
- OpenJDK 64-Bits 20.0.2.
- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- Herencia y Polimorfismo

### 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/fase02/lab20>

### 4. Actividades con el repositorio GitHub

#### 4.1. Preparación del espacio de trabajo

Listing 1: Se crea la carpeta de laboratorio 20 y se copian los archivos del lab012 al lab20

```
mkdir lab20
cd lab20
Copy-Item "Soldier.java" -Destination "..\lab20"
Copy-Item "VideoJuego5.java" -Destination "..\lab12\VideoJuego.java"
cd ..
cd lab20
code .
```

Listing 2: Commit: 99f16a3c517c18d8206b100d33722e5805398f60

```
git add Soldier.java
git commit -m "Se copia las clase Soldier.java y VideoJuego.java del lab12 al 20"
git push -u origin main
```

#### 4.2. Aplicación de la Herencia

Listing 3: Se modifican metodos de la clase Soldier

code Soldier.java

```
193     public String toString() {
194         return "name=" + name + ", row=" + row + ", column=" + column + ", attackLevel=" + attackLevel
195             + ", defenseLevel=" + defenseLevel + ", actualLife=" + actualLife + ", speed=" + speed + ", attitude="
196             + attitude + ", current=" + current;
197     }
```

- Como las demás clases heredarán de Soldier sus métodos y atributos, para una mejor visualización de los datos de la clase, se corrige la forma en la que el toString de la versión anterior mostraba

Listing 4: Se crea la subclase Archer

code Archer.java

```
1 public class Archer extends Soldier {
2     private int arrows;
3
4     public Archer(int arrows) {
5         this.arrows = arrows;
6     }
7
8     public Archer(String name, int row, int column, int attackLevel, int defenseLevel, int actualLife, int speed,
9         String attitude, boolean current, int arrows) {
10         super(name, row, column, attackLevel, defenseLevel, actualLife, speed, attitude, current);
11         this.arrows = arrows;
12     }
13
14     public Archer(String name, int row, int column, int attackLevel, int defenseLevel, int actualLife, int speed,
15         int arrows) {
16         super(name, row, column, attackLevel, defenseLevel, actualLife, speed);
17         this.arrows = arrows;
18     }
19
20     public void shoot() {
21         if (arrows > 0) {
22             arrows--;
23         }
24     }
25
26     @Override
27     public String toString() {
28         return "[Archer: " + super.toString() + ", arrows=" + arrows + "]";
29     }
30
31 }
```

Listing 5: Commit: c3fab4bac8b4e5e7a1463ecc8e1f205b3f755977

```
git add Archer.java
git commit -m "Se crea el metodo shoot y el toString"
git push -u origin main
```

Listing 6: Se crea la subclase Knight

```
code Knight.java
```

```
30     public void envestir() {
31         if (horseRiding == true)
32             for (int i = 0; i <= 2; i++)
33                 super.attack();
34         else
35             super.attack();
36         super.attack();
37     }
38
39     public void desmontar() {
40         if (horseRiding == true) {
41             horseRiding = false;
42             super.defend();
43             cambiaArma();
44         }
45     }
46
47     public void cambiaArma() {
48         if (currentWeapon == "Lanza")
49             currentWeapon = "Espada";
50         else
51             currentWeapon = "Lanza";
52     }
53
54     public void montar() {
55         if (horseRiding == false) {
56             horseRiding = true;
57             super.attack();
58             cambiaArma();
59         }
60     }
```

- Este fragmento es lo más interesante de esta clase y lo que la práctica de laboratorio específica

Listing 7: Commit: a4b277fea2814d4543b4b906fc545da59c009198

```
git add Knight.java
git commit -m "Se agrega la funcion attack dentro de investir"
git push -u origin main
```

Listing 8: Se crea la subclase Spearman

```
code Spearman.java
```

```
22     public void schiltrom() {
23         int newDefense = getDefenseLevel() +1;
24         setDefenseLevel(newDefense);
25     }
```

- Este fragmento es lo más interesante de esta clase y lo que la práctica de laboratorio específica

Listing 9: Commit: 2039a89dc9095c3a2a7acd26f64f5efeddae5e23

```
git add Spearman.java
git commit -m "Se implementa el metodo schiltrom"
git push -u origin main
```

Listing 10: Se crea la subclase Swordsman

```
code Swordsman.java
```

```
22     public void muroEscudos() {
23         if(shieldWalls==true) {
24             shieldWalls = false;
25         } else {
26             shieldWalls = true;
27         }
28     }
```

- Este fragmento es lo más interesante de esta clase y lo que la práctica de laboratorio específica

Listing 11: Commit: 82b0ba62ff23779dc047659700c9991032678c3b

```
git add Swordsman.java
git commit -m "Se crea el metodo muroEscudos y el toString se sobrescribe"
git push -u origin main
```

### 4.3. Aplicación de Composición

Listing 12: Se crea la Army

code Army.java

```
12 public Army(String name) {
13     ArrayList<ArrayList<Soldier>> army = new ArrayList<>(initialCapacity:10);
14     for (int i = 0; i < 10; i++) {
15         army.add(new ArrayList<>(Collections.nCopies(n:10, o:null)));
16     }
17     this.army = army;
18     this.name = name;
19 }
```

- Este fragmento ya establece a los atributos ArrayList valores null y establece el nombre de cada ejercito que se creó

Listing 13: Commit: 2f415f95bd8470ae0b16e5a2c7af7d2fdfb1a841

```
git add Army.java
git commit -m "Se crean 3 constructores sobrecargados"
git push -u origin main
```

Listing 14: Se crea generateArmy

code Army.java

```

30 // 0:Archer; 1:Knight; 2:Spearman; 3:Swordsman
31 public Army generateArmy(Army a) {
32     Random random = new Random();
33     int amount = random.nextInt(bound:10) + 1;
34     int n = 0;
35     int row, column, attack, defense, lifePoints;
36     String name;
37     int optionSoldier;
38     Soldier sol;
39     do {
40         row = random.nextInt(ROW_BOARD);
41         column = random.nextInt(COLUMN_BOARD);
42         if (a.getSoldier(row, column) == null && a.get(row).get(column) == null) {
43
44             optionSoldier = random.nextInt(bound:4);
45             if (optionSoldier == 0) {
46                 attack = 7;
47                 defense = 3;
48                 lifePoints = random.nextInt(bound:5) + 3;
49                 name = "Archer" + (row + 1) + "x" + (column + 1);
50                 sol = new Archer(name, row + 1, column + 1, attack, defense, lifePoints, speed:0, arrows:5);
51
52             } else if (optionSoldier == 1) {
53                 attack = 13;
54                 defense = 7;
55                 lifePoints = random.nextInt(bound:3) + 10;
56                 name = "Knight" + (row + 1) + "x" + (column + 1);
57                 sol = new Knight(name, row + 1, column + 1, attack, defense, lifePoints, speed:0, currentWeapon:"sword", horseRiding:false);
58
59             } else if (optionSoldier == 2) {
60                 attack = 5;
61                 defense = 10;
62                 lifePoints = random.nextInt(bound:4) + 5;
63                 name = "Spearman" + (row + 1) + "x" + (column + 1);
64                 sol = new Spearman(name, row + 1, column + 1, attack, defense, lifePoints, speed:0, longLance:1);
65
66             } else {
67                 attack = 10;
68                 defense = 8;
69                 lifePoints = random.nextInt(bound:3) + 8;
70                 name = "Swordsman" + (row + 1) + "x" + (column + 1);
71                 sol = new Swordsman(name, row + 1, column + 1, attack, defense, lifePoints, speed:0, longSword:1, shieldWalls:false);
72             }
73             a.setSoldier(row, column, a, sol);
74             n++;
75         }
76     } while (n < amount);
77     return a;
78 }

```

- Este método guarda la misma lógica para generar el ejército del lab12, solo que aquí se nos pide incluir dentro del ejército soldados que son de la clase hija, por lo que se hace uso del Polimorfismo. Para agregar los soldados de las clases hija se hace de forma aleatoria

Listing 15: Commit: cd5e2d1a4c71b2db3ae28978a840f7c4ff318738

```

git add Army.java
git commit -m "Se termina de generar soldados de todos los tipos por ejercito"
git push -u origin main

```

Listing 16: Se crean métodos que nos permitirán acceder a los atributos de la clase

code Army.java

```

95     public void showArmy() {
96         for (ArrayList<Soldier> row : army) {
97             for (Soldier soldier : row) {
98                 if (soldier != null) {
99                     System.out.println(soldier);
100                 }
101             }
102         }
103     }
104
105     public ArrayList<ArrayList<Soldier>> getArmyInArrayListBi() {
106         return army;
107     }
108
109     public ArrayList<Soldier> getArmyInArrayUni() {
110         return armyU;
111     }
112

```

- Estos métodos nos permiten visualizar los soldier de cada ejército y obtener los atributos que son ArrayList. Antes de estos métodos venía otro el cual convertía ArrayList Bidimensionales en Unidimensionales, prácticamente se copió este método de la clase VideoJuego a la clase Army

Listing 17: Commit: 0f87b3ab9106b3b59a30c056e21495671d0e5fa2

```

git add Army.java
git commit -m "Se crean metodos que nos serviran para acceder al atributo que es de
tipo ArraList de la clase Army y tambien podremos visualizar los soldier de cada ejercito"

git push -u origin main

```

Listing 18: Se termina de crear los demás métodos

```
code Army.java
```



```

113 // 0:archer 1:Knight 2:Spearman 3:Swordsman 5:total
114 // En la guía se nos pide contabilizar los objetos
115 public int[] count() {
116     converterToArrayUni();
117     int[] num = new int[5];
118     int total = 0;
119     for (Soldier sol : armyU) {
120         if (sol instanceof Archer) {
121             num[0]++;
122         } else if (sol instanceof Knight) {
123             num[1]++;
124         } else if (sol instanceof Spearman) {
125             num[2]++;
126         } else {
127             num[3]++;
128         }
129     }
130     for (int n : num) {
131         total = total + n;
132     }
133     num[4] = total;
134
135     System.out.print("Archer: " + num[0] + " ");
136     System.out.print("Knight: " + num[1] + " ");
137     System.out.print("Spearman: " + num[2] + " ");
138     System.out.print("Swordsman: " + num[3] + " ");
139     System.out.print("Total: " + num[4] + " ");
140     return num;
141 }
142
143 // ordena un arreglo bidimensional
144 public void organize() {
145     int n = armyU.size();
146     for (int i = 1; i < n; i++) {
147         Soldier key = armyU.get(i);
148         int j = i - 1;
149         while (j >= 0 && armyU.get(j).getActualLife() > key.getActualLife()) {
150             armyU.set(j + 1, armyU.get(j));
151             j--;
152         }
153         armyU.set(j + 1, key);
154     }
155 }

```

- Se crearon otros métodos pero en su mayoría solo son accesorios y que básicamente necesitan de estos dos para poder funcionar. Count nos devuelve la cantidad de soldados y también hace la contabilización de las subclases hijas. En cuanto al método Organize usa un método de ordenamiento, este método ya se vino trabajando en laboratorios pasados, solo que en esta versión se le adapta para que sea un método de clase y no estático como lo era antes

Listing 19: Commit: 0f87b3ab9106b3b59a30c056e21495671d0e5fa2

```
git add Board.java
git commit -m "Se crean metodos que nos sirvan para acceder al atributo que es de
tipo ArrayList de la clase Army y tambien podremos visualizar los soldier de cada
ejercito"
git push -u origin main
```

#### 4.4. Clase Board

Listing 20: Se crean los métodos que generan caracteres para el tablero

```
code Board.java
```

```

26 // 0:Bosque 1:Campo 2:Montania 3:Desierto 4: Playa
27 public String chooseBattleField() {
28     Random random = new Random();
29     int option = random.nextInt(bound:5);
30     if (option == 0) {
31         typeTerritory = "Bosque";
32     } else if (option == 1) {
33         typeTerritory = "Campo";
34     } else if (option == 2) {
35         typeTerritory = "Montania";
36     } else if (option == 3) {
37         typeTerritory = "Desierto";
38     } else {
39         typeTerritory = "Playa";
40     }
41     return typeTerritory;
42 }
43
44 public static String kingdom(String str) {
45     if (str.equalsIgnoreCase(anotherString:"Castilla")) {
46         return "C";
47     } else if (str.equalsIgnoreCase(anotherString:"Francia")) {
48         return "F";
49     } else if (str.equalsIgnoreCase(anotherString:"Moros")) {
50         return "M";
51     } else if (str.equalsIgnoreCase(anotherString:"Inglaterra")) {
52         return "I";
53     } else {
54         return "R";
55     }
56 }
57
58 public String battlefield() {
59     String charMap;
60     if (typeTerritory.equalsIgnoreCase(anotherString:"playa")) {
61         return "P";
62     } else if (typeTerritory.equalsIgnoreCase(anotherString:"montañana")) {
63         return "M";
64     } else if (typeTerritory.equalsIgnoreCase(anotherString:"desierto")) {
65         return "D";
66     } else if (typeTerritory.equalsIgnoreCase(anotherString:"Campo")) {
67         return "C";
68     } else {
69         return "B";
70     }
71 }
72
73 public String typSoldier(Soldier sol) {
74     if (sol instanceof Archer) {
75         return "a";
76     } else if (sol instanceof Knight) {
77         return "r";
78     } else if (sol instanceof Spearman) {
79         return "l";
80     } else {
81         return "e";
82     }
83 }

```

- Por supuesto que antes de crear estos métodos ya se han creado los atributos y el constructor. El constructor recibe como parámetros dos objetos de clase Army (se evidencia aquí más el uso de la composición). En cuanto a los métodos de la presente imagen, se limitan a poder establecer

los caracteres que se imprimirán en el tablero

Listing 21: Commit: 4c61c72b558e8335ed0dda950c796c2caaa121e0

```
git add Board.java
git commit -m "Se crean metodos que generan un caracter para el tablero"
git push -u origin main
```

Listing 22: Se modifica la impresión del tablero

code Board.java

```

94     public void generateMap() {
95
96         ArrayList<ArrayList<Soldier>> a = army1.getArmyInArrayListBi();
97         ArrayList<ArrayList<Soldier>> b = army2.getArmyInArrayListBi();
98
99         String charMap = battlefield();
100         map = new String[ROW][COLUMN];
101
102         for (int i = 0; i < map.length; i++) {
103             for (int j = 0; j < map[i].length; j++) {
104                 map[i][j] = "|_|_|_|_|_|_|_|";
105             }
106         }
107
108         for (int i = 0; i < a.size(); i++) {
109             for (int j = 0; j < a.get(i).size(); j++) {
110                 if (a.get(i).get(j) != null) {
111                     String strA = "|_" + charMap + "*" + Board.kingdom(kingdom1) + typSoldier(a.get(i).get(j))
112                         + lifeInStringFormat(i, j, a) + "*" + charMap + "_|";
113                     map[i][j] = strA;
114                 }
115             }
116         }
117         for (int i = 0; i < b.size(); i++) {
118             for (int j = 0; j < b.get(i).size(); j++) {
119                 if (b.get(i).get(j) != null && map[i][j] != "s") {
120                     String strB = "|_" + charMap + "*" + Board.kingdom(kingdom2) + typSoldier(b.get(i).get(j))
121                         + lifeInStringFormat(i, j, b) + "*" + charMap + "_|";
122                     map[i][j] = strB;
123                 }
124             }
125         }
126         System.out.print(
127             s:"      A      B      C      D      E      F      G      H
128         for (int i = 0; i < map.length; i++) {
129             System.out.printf(format:"%2d", (i + 1));
130             for (int j = 0; j < map[i].length; j++) {
131                 System.out.print(map[i][j]);
132             }
133             System.out.println();
134         }
135     }

```

- La lógica con la que se genera el tablero no ha cambiado, solo se hacen leves modificaciones como la inclusión de nuevos caracteres que simulan el tipo de arena, el reino, el tipo o clase de soldado que es, su valor de vida actual y por ultimo se vuelve a poner el carácter que simula el ambiente del combate

Listing 23: Commit: ea7d152254cd6643d1efa6457507b9b3f09e5c5f

```
git add Board.java
git commit -m "Se modifica el metodo myBoard que era estatico, ahora es un metodo de
               clase y tambien se le cambio el nombre al metodo"
git push -u origin main
```

## 4.5. Clase GameFast

Listing 24: Se crea la clase que gestiona el juego

code GameFast.java

```

9  public GameFast(Board board, Army army1, Army army2, boolean gameOver) {
10     this.board = board;
11     this.army1 = army1;
12     this.army2 = army2;
13     this.gameOver = gameOver;
14 }
15
16 public void winner() {
17     String kingdom1 = army1.getName();
18     String kingdom2 = army2.getName();
19
20     Random random = new Random();
21     int definitiveProbability = random.nextInt(bound:100);
22
23     int total1 = totallife(army1);
24     int total2 = totallife(army2);
25
26     double totallife = total1 + total2;
27     double probabilityAttack1 = 100 * total1 / totallife;
28     double probabilityAttack2 = 100 - probabilityAttack1;
29
30     System.out.println("Estadísticas de batalla:\tReino " + kingdom1 + ": " + probabilityAttack1
31         + "%\tReino " + kingdom2 + ": " + probabilityAttack2 + "%\tSalio como aleatorio: "
32         + definitiveProbability + "%");
33
34     if (definitiveProbability <= probabilityAttack1) {
35         System.out.println("Gano el reino " + kingdom1);
36     } else if (definitiveProbability <= probabilityAttack2) {
37         System.out.println("Gano el reino " + kingdom2);
38     } else {
39         System.out.println(x:"Empate");
40     }
41 }
42
43 public int totallife(Army army) {
44     int sumLife = 0;
45     for (Soldier sol : army.getArmyInArrayUni()) {
46         sumLife = sumLife + sol.getActuallife();
47     }
48     return sumLife;
49 }

```

- En esta imagen solo se verá dos métodos que son importantes, el primero es winner que cual dictamina al ganador mediante el criterio de probabilidades, por ejemplo si un ejército sacó 30

por ciento, mientras que el otro 70 y la probabilidad definitiva es es 23 por ciento, el primer ejército gana ya que está dentro de su rango. Sin embargo, para que este método pueda hacer su función requiere la suma total de los puntos de vida de cada army, por lo que se crea el método totalLife el cual calcula ello.

Listing 25: Commit: 289acd859314fec76473faa1f7adbae3f3de7c0

```
git add GameFast.java
git commit -m "Se crean los metodos winner y totalLife de GameFast"
git push -u origin main
```

#### 4.6. Clase Aplicación y ejecución del programa

Listing 26: Se crea la clase principal que llama a todas las clases

```
code Aplicacion.java
```



```

6  import java.util.*;
7
8  public class Aplicacion {
9      private static final Scanner sc = new Scanner(System.in);
10
11      Run | Debug
12      public static void main(String[] args) {
13          String kingdom1;
14          String kingdom2;
15
16          while (validation()) {
17              System.out.println(x: "Elija su reino (Francia, Inglaterra, Moros, Roma, Castilla)");
18              kingdom1 = sc.next();
19
20              System.out.println(x: "Elija su reino (Francia, Inglaterra, Moros, Roma, Castilla)");
21              kingdom2 = sc.next();
22
23              Army a = new Army(kingdom1);
24              Army b = new Army(kingdom2);
25
26              a.generateArmy(b);
27              b.generateArmy(a);
28
29              a.showArmy();
30              System.out.println();
31              b.showArmy();
32              System.out.println();
33
34              Board tablero = new Board(a, b, kingdom1: "Castilla", kingdom2: "Francia");
35              tablero.generateMap();
36
37              System.out.println(x: "Contando los soldados");
38              System.out.println("Ejercito 1:" + a.count());
39              System.out.println("Ejercito 2:" + b.count());
40              System.out.println();
41
42              System.out.println(x: "Datos de los soldados con mayor vida");
43              a.longerLife();
44              b.longerLife();
45
46              System.out.println();
47              GameFast gameFast = new GameFast(tablero, a, b, gameOver: false);
48              gameFast.winner();
49          }
50      }

```

- Básicamente esta clase es la principal ya que se encarga de la ejecución de todo el programa. Se hace un juego iterativo.

```

52  public static boolean validation() {
53      do {
54          System.out.println(x:"Desea jugar una ronda?(si/no)");
55          String answer = sc.next();
56          if (answer.equalsIgnoreCase(anotherString:"Si")) {
57              return true;
58          } else if (answer.equalsIgnoreCase(anotherString:"No")) {
59              return false;
60          } else {
61              System.out.println(x:"Respuesta no admsible");
62          }
63      } while (true);
64  }
65  }

```

- Adicionalmente se implementa un método el cual se encargará de recepcionar si el usuario quiere seguir jugando. Este método es de laboratorios pasados.

Listing 27: Commit: de043b03d19c52fc136cf3167d29a847991b6a47

```

git add Aplicacion.java
git commit -m "Se crea la clase aplicacion que llama a todos las clases"
git push -u origin main

```

Listing 28: Compilando y probando todo el juego

```

Desea jugar una ronda?(si/no)
si
Elija su reino (Francia, Inglaterra, Moros, Roma, Castilla)
Francia
Elija su reino (Francia, Inglaterra, Moros, Roma, Castilla)
Castilla
[Archer: name=Archer4x8, row=4, column=8, attackLevel=7, defenseLevel=3, actualLife=3,
speed=0, attitude=Repose, current=true, arrows=5]
Swordsman: name=Swordsman6x1, row=6, column=1, attackLevel=10, defenseLevel=8,
actualLife=8, speed=0, attitude=Repose, current=true, longSnow=1.0, shieldWalls=false]
Spearman: name=Spearman7x1, row=7, column=1, attackLevel=5, defenseLevel=10, actualLife=6,
speed=0, attitude=Repose, current=true, lonLance=1]
Spearman: name=Spearman8x9, row=8, column=9, attackLevel=5, defenseLevel=10, actualLife=6,
speed=0, attitude=Repose, current=true, lonLance=1]

[Knight: name=Knight3x1, row=3, column=1, attackLevel=13, defenseLevel=7, actualLife=10,
speed=0, attitude=Repose, current=true, currentWeapon=sword, horseRiding=false]
Spearman: name=Spearman3x3, row=3, column=3, attackLevel=5, defenseLevel=10, actualLife=7,
speed=0, attitude=Repose, current=true, lonLance=1]
[Archer: name=Archer4x1, row=4, column=1, attackLevel=7, defenseLevel=3, actualLife=3,
speed=0, attitude=Repose, current=true, arrows=5]
Swordsman: name=Swordsman4x4, row=4, column=4, attackLevel=10, defenseLevel=8,
actualLife=9, speed=0, attitude=Repose, current=true, longSnow=1.0, shieldWalls=false]
Spearman: name=Spearman6x2, row=6, column=2, attackLevel=5, defenseLevel=10, actualLife=7,
speed=0, attitude=Repose, current=true, lonLance=1]

```



```
Swordsman: name=Swordsman9x6, row=9, column=6, attackLevel=10, defenseLevel=8,
    actualLife=9, speed=0, attitude=Repose, current=true, longSnow=1.0, shieldWalls=false]
Spearman: name=Spearman9x10, row=9, column=10, attackLevel=5, defenseLevel=10,
    actualLife=6, speed=0, attitude=Repose, current=true, lonLance=1]
Spearman: name=Spearman10x3, row=10, column=3, attackLevel=5, defenseLevel=10,
    actualLife=8, speed=0, attitude=Repose, current=true, lonLance=1]
```

A	B	C	D	E	F	G	H	I
		J						
1	_	_	_	_	_	_	_	_
2	_	_	_	_	_	_	_	_
3	_C*Fr10*C_	_	_C*F107*C_	_	_	_	_	_
4	_C*Fa03*C_	_	_	_C*Fe09*C_	_	_	_	_C*Ca03*C_
5	_	_	_	_	_	_	_	_
6	_C*Ce08*C_	_C*F107*C_	_	_	_	_	_	_
7	_C*Cl06*C_	_	_	_	_	_	_	_
8	_	_	_	_	_	_	_	_C*Cl06*C_
9	_	_	_	_	_	_C*Fe09*C_	_	_
10	_	_	_C*F108*C_	_	_	_	_	_

Contando los soldados

Archer: 1 Knight: 0 Spearman: 2 Swordsman: 1 Total: 4 Ejercito 1:[I@41906a77

Archer: 1 Knight: 1 Spearman: 4 Swordsman: 2 Total: 8 Ejercito 2:[I@4b9af9a9

Datos de los soldados con mayor vida

```
Swordsman: name=Swordsman6x1, row=6, column=1, attackLevel=10, defenseLevel=8,
    actualLife=8, speed=0, attitude=Repose, current=true, longSnow=1.0, shieldWalls=false]
[Knight: name=Knight3x1, row=3, column=1, attackLevel=13, defenseLevel=7, actualLife=10,
    speed=0, attitude=Repose, current=true, currentWeapon=sword, horseRiding=false]
```

Estadísticas de batalla: Reino Francia: 28.048780487804876% Reino Castilla:  
71.95121951219512% Salio como aleatorio: 97%

Empate

Desea jugar una ronda?(si/no)

no

- Pido disculpas por la presentación del tablero y los datos de los soldados, ya son problemas de Latex y no tanto de mi código

[illegible]

- #### 4.8. Estructura de laboratorio 20

- Página 18

```
C:.\
|
|  Aplicacion.java
|  Archer.java
|  Army.java
|  Board.java
|  GameFast.java
|  Knight.java
|  Soldier.java
|  Spearman.java
|  Swordsman.java
|
|_ latex
|   programacion_lab20_rescobedoq_v1.0.pdf
|   programacion_lab20_rescobedoq_v1.0.tex
|
|_ img
|   archer.jpg
|   binary.jpg
|   burbuja.jpg
|   chars.jpg
|   classSoldier.jpg
|   constructorArmy.jpg
|   gameFast.jpg
|   generateArmy.jpg
|   getArmy.jpg
|   insertion.jpg
|   knight.png
|   logo_abet.png
|   logo_episunsa.png
|   logo_unsa.jpg
|   main.jpg
|   muro.jpg
|   otherArmy.jpg
|   printBoard.jpg
|   spearman.jpg
|   toString.jpg
|   uml.png
|   validation.jpg
```

- Debido a problemas en la codificación de latex con ciertos caracteres que genera el comando tree

f, se opta por enviar la estructura de este laboratorio en formato de imagen

## 5. Rúbricas

### 5.1. Entregable Informe

Tabla 1: Tipo de Informe

<b>Informe</b>	
<b>Latex</b>	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 cumple 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 los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

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

Tabla 3: Rúbrica para contenido del Informe y demostración

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

## 6. Referencias

- <https://www.geeksforgeeks.org/insertion-sort/>