Código Tarea Final

Clase Flow

```
public class Flow {

  public static void main(String[] args) {

    // Equipos

    Equipo atleti = new Equipo("Atletico de Madrid", "Cívitas Metropolitano",
        new Jugador[] { new Jugador("Oblak", 13, "POR", 3, 87), new Jugador("Savic", 15, "DEF", 12, 32),
            new Jugador("Witsel", 20, "DEF", 46, 38), new Jugador("Hermoso", 22, "DEF", 18, 28),
            new Jugador("M.Llorente", 14, "MED", 61, 36), new Jugador("R.De Paul", 5, "MED", 65, 30),
            new Jugador("Koke", 6, "MED", 40, 15), new Jugador("Pablo Barrios", 24, "MED", 70, 20),
            new Jugador("A.Correa", 10, "DEL", 85, 8), new Jugador("A.Griezmann", 7, "DEL", 92, 10),
            new Jugador("Moratatatata", 19, "DEL", 75, 7) });

    Equipo realMadrid = new Equipo("Real Madrid", "Santiago Bernabéu",
        new Jugador[] { new Jugador("Kepa", 1, "POR", 3, 83), new Jugador("Carvajal", 2, "DEF", 37, 38),
            new Jugador("Rudiger", 22, "DEF", 60, 33), new Jugador("Nacho", 6, "DEF", 12, 24),
            new Jugador("F.Mendy", 23, "DEF", 12, 20), new Jugador("T.Kroos", 8, "MED", 67, 15),
            new Jugador("L.Modric", 10, "MED", 74, 13), new Jugador("Valverde", 15, "MED", 78, 36),
            new Jugador("Bellingham", 5, "MED", 90, 35), new Jugador("Rodrygo", 11, "DEL", 84, 5),
            new Jugador("Joselu", 14, "DEL", 82, 5) });

    Equipo barsa = new Equipo("FC Barcelona", "Spotify Camp Nou",
        new Jugador[] { new Jugador("Ter Stegen", 1, "POR", 3, 91), new Jugador("Cancelo", 2, "DEF", 37, 38),
            new Jugador("Kounde", 23, "DEF", 60, 33), new Jugador("Araujo", 4, "DEF", 12, 24),
            new Jugador("A.Balde", 23, "DEF", 12, 20), new Jugador("F.De Jong", 21, "MED", 40, 25),
            new Jugador("Pedri", 10, "MED", 60, 15), new Jugador("Gundogan", 22, "MED", 65, 10),
            new Jugador("Raphinha", 11, "DEL", 78, 12), new Jugador("Joao Félix", 14, "DEL", 82, 12),
            new Jugador("Lewandowski", 9, "DEL", 91, 5) });

    Equipo city = new Equipo("Manchester City", "Ciudad de Mánchester",
        new Jugador[] { new Jugador("Ederson", 31, "POR", 3, 88), new Jugador("Walker", 2, "DEF", 37, 38),
            new Jugador("Ruben Dias", 3, "DEF", 8, 40), new Jugador("N.Aké", 6, "DEF", 10, 22),
            new Jugador("Rodri", 16, "MED", 67, 36), new Jugador("Bernardo Silva", 20, "MED", 76, 12),
            new Jugador("Akanji", 25, "MED", 15, 24), new Jugador("Doku", 11, "DEL", 77, 10),
            new Jugador("P.Foden", 11, "DEL", 78, 12), new Jugador("Julián Alvarez", 19, "DEL", 81, 8),
            new Jugador("E.Haaland", 9, "DEL", 94, 3) });

    Equipo liverpool = new Equipo("Liverpool", "Anfield",
        new Jugador[] { new Jugador("Alisson", 1, "POR", 3, 89),
            new Jugador("T.Alexander-Arnold", 66, "DEF", 21, 34),
            new Jugador("V.Van Dijk", 3, "DEF", 8, 50), new Jugador("Matip", 32, "DEF", 8, 30),
            new Jugador("Robertson", 26, "DEF", 13, 30), new Jugador("Ryan Gravenberch", 38, "MED", 66, 21),
            new Jugador("Mac Allister", 10, "MED", 65, 20), new Jugador("Szoboszlai", 8, "MED", 75, 10),
            new Jugador("Luis Diaz", 7, "DEL", 83, 14), new Jugador("Mohamed Salah", 11, "DEL", 92, 8),
            new Jugador("Darwin Nuñez", 9, "DEL", 82, 3) });

    Equipo bayern = new Equipo("FC Bayern", "Allianz Arena",
        new Jugador[] { new Jugador("Neuer", 1, "POR", 3, 86), new Jugador("Mazraoui", 40, "DEF", 12, 30),
            new Jugador("Upamecano", 2, "DEF", 4, 35), new Jugador("Kim Min Jae", 3, "DEF", 11, 33),
            new Jugador("A.Davies", 26, "DEF", 13, 32), new Jugador("Laimer", 27, "MED", 54, 34),
            new Jugador("L.Goretzka", 8, "MED", 67, 22), new Jugador("K.Coman", 11, "DEL", 81, 10),
            new Jugador("J.Musiala", 7, "MED", 75, 16), new Jugador("Leroy Sane", 10, "DEL", 80, 8),
            new Jugador("Harry Kane", 9, "DEL", 93, 3) });

    Equipo milan = new Equipo("AC Milan", "San Siro",
        new Jugador[] { new Jugador("Maignan", 16, "POR", 3, 84), new Jugador("Calabria", 2, "DEF", 9, 23),
            new Jugador("Tomori", 23, "DEF", 7, 41), new Jugador("P.Kalulu", 20, "DEF", 12, 33),
            new Jugador("Theo Hernandez", 19, "DEF", 38, 35), new Jugador("T.Reijnders", 14, "MED", 57, 31),
            new Jugador("Y.Musah", 80, "MED", 51, 25), new Jugador("C.Pulisic", 11, "MED", 78, 11),
            new Jugador("Loftus-Cheek", 8, "MED", 80, 24), new Jugador("Rafael Leao", 10, "DEL", 85, 8),
            new Jugador("O.Giroud", 9, "DEL", 88, 2) });

    Equipo psg = new Equipo("PSG", "Parque De Los Príncipes",
```

```
        new Jugador[] { new Jugador("Donnarumma", 99, "POR", 3, 91), new Jugador("A.Hakimi", 2, "DEF", 19, 25),
            new Jugador("Marquinhos", 5, "DEF", 12, 35), new Jugador("Skriniar", 37, "DEF", 8, 33),
            new Jugador("Lucas Hernandez", 21, "DEF", 22, 29), new Jugador("Ugarte", 4, "MED", 24, 28),
            new Jugador("Vitinha", 17, "MED", 26, 19), new Jugador("Lee Kang-in", 19, "MED", 70, 16),
            new Jugador("O.Dembele", 10, "DEL", 84, 8), new Jugador("K.Mbappe", 7, "DEL", 93, 4),
            new Jugador("Kolo Muani", 23, "DEL", 84, 5) });

    Partido cuartosA = new Partido(bayern, realMadrid);
    Partido cuartosB = new Partido(barsa, milan);
    Partido cuartosC = new Partido(psg, city);
    Partido cuartosD = new Partido(atleti, liverpool);

    System.out.println("\n\n---------------- CUARTOS DE FINAL ----------------");
    Partido semis1 = new Partido(cuartosA.jugarPartido(), cuartosC.jugarPartido());
    Partido semis2 = new Partido(cuartosB.jugarPartido(), cuartosD.jugarPartido());

    System.out.println("\n\n---------------- SEMIFINALES ----------------");
    Partido finalChampions = new Partido(semis1.jugarPartido(), semis2.jugarPartido());

    System.out.println("\n\n---------------- FINAL DE LA CHAMPIONS ----------------");
    System.out
        .println("\nEl ganador de la Champions es " + finalChampions.jugarPartido().getNombre().toUpperCase());

  }

}
```

Clase Partido

```
public class Partido {

  // Atributos del partido

  private Equipo local, visitante;

  private final int TIEMPOFINAL = 90;
  private final int MEDIOTIEMPO = TIEMPOFINAL / 2;
  private final int PRORROGA = 120;
  private int minuto = 1;

  private int golesLocal = 0;
  private int golesVisitante = 0;

  private final int PENALTIS = 5;
  private int tirador = 0;

  private int ronda = 1;
  private int penaltisLocal = 0;
  private int penaltisVisitante = 0;

  // Constructores (Overloading)

  public Partido() {
    this.local = null;
    this.visitante = null;
  }

  public Partido(Equipo local, Equipo visitante) {
    this.local = local;
    this.visitante = visitante;
  }

  // Método principal

  public Equipo jugarPartido() {

    System.out.println("\n\n-------- " + local.getNombre() + " VS " + visitante.getNombre() + " --------");
```

```
  System.out.println(local.toString());

  System.out.println(visitante.toString());

  esperaLenta();

  System.out.println("\n---------------- PRIMER TIEMPO ----------------");

  for (minuto = 1; minuto <= TIEMPOFINAL; minuto++) {

    if (aleatoriedad()) {
      jugadaLocal();
    } else {
      jugadaVisitante();
    }

    espera();

    if (minuto == MEDIOTIEMPO) {
      espera();
      System.out.println("\n---------------- SEGUNDO TIEMPO ----------------");
    }

  }

  esperaLenta();

  if (golesLocal == golesVisitante) {

    System.out.println("\n---------------- PRORROGA ----------------");

    for (; minuto <= PRORROGA; minuto++) {

      espera();

      if (aleatoriedad()) {
        jugadaLocal();
      } else {
        jugadaVisitante();
      }

    }

    if (golesLocal == golesVisitante) {
      System.out.println("\n---------------- PENALTIS ----------------");
      penaltis(aleatoriedad());
    } else {
      System.out.println("\n---------------- RESULTADO FINAL ----------------\n" + local.getNombre() + " "
          + golesLocal + " - " + golesVisitante + " " + visitante.getNombre());
    }

  } else {
    System.out.println("\n---------------- RESULTADO FINAL ----------------\n" + local.getNombre() + " "
        + golesLocal + " - " + golesVisitante + " " + visitante.getNombre());
  }

  esperaLenta();

  return (golesLocal + penaltisLocal > golesVisitante + penaltisVisitante) ? local : visitante;

}

// Métodos de tiros a puerta

private void jugadaLocal() {

  for (Jugador jugador : local.getConvocatoria()) {
    if (tiro(jugador.getTiro()) && !parada(visitante)) {
      golesLocal++;
      System.out.println(jugador.getNombre() + " " + minuto + "\'" + (isPenalti() ? " (P)" : "") + " ("
          + local.getNombre() + ")");
      espera();
```

```
      break;
    }
  }

}

private void jugadaVisitante() {

  for (Jugador jugador : visitante.getConvocatoria()) {
    if (tiro(jugador.getTiro()) && !parada(local)) {
      golesVisitante++;
      System.out.println(jugador.getNombre() + " " + minuto + "\'" + (isPenalti() ? " (P)" : "") + " ("
          + visitante.getNombre() + ")");
      espera();
      break;
    }
  }

}

// Método de la fase de penaltis

private void penaltis(boolean moneda) {
  System.out.println("\n" + local.getNombre() + " " + golesLocal
      + ((penaltisLocal != 0) ? "(" + penaltisLocal + ") - " : " - ") + golesVisitante
      + ((penaltisVisitante != 0) ? "(" + penaltisVisitante + ") - " : " ") + visitante.getNombre());

  esperaLenta();

  System.out.println("\n---------------- RONDA " + ronda + " ----------------");

  for (int i = 0; i < PENALTIS; i++) {

    tirador--;

    if (tirador < 0) {
      tirador = Math.min(local.getConvocatoria().length, visitante.getConvocatoria().length) - 1;
    }

    esperaLenta();

    if (moneda) {
      penaltiLocal(tirador);
      esperaLenta();
      penaltiVisitante(tirador);
    } else {
      penaltiVisitante(tirador);
      esperaLenta();
      penaltiLocal(tirador);
    }

    if ((PENALTIS - 1) - i + penaltisLocal < penaltisVisitante
        || (PENALTIS - 1) - i + penaltisVisitante < penaltisLocal) {
      break;
    }

  }

  ronda++;

  if (penaltisLocal != penaltisVisitante) {
    esperaLenta();
    System.out.println("\n---------------- RESULTADO FINAL ----------------\n" + local.getNombre() + " "
        + golesLocal + "(" + penaltisLocal + ") - " + golesVisitante + "(" + penaltisVisitante + ") "
        + visitante.getNombre());
  } else {
    penaltis(moneda);
  }

}

// Métodos de tiros en penaltis
```

```
  private void penaltiLocal(int tirador) {

    if (penalti(local.getConvocatoria()[tirador].getTiro()) && !paradaPenalti(visitante)) {
      penaltisLocal++;
      System.out.println(
          "Goool de " + local.getConvocatoria()[tirador].getNombre() + " (" + local.getNombre() + ")");
    } else {
      System.out.println(
          "Fallo de " + local.getConvocatoria()[tirador].getNombre() + " (" + local.getNombre() + ")");
    }

  }

  private void penaltiVisitante(int tirador) {

    if (penalti(visitante.getConvocatoria()[tirador].getTiro()) && !paradaPenalti(local)) {
      penaltisVisitante++;
      System.out.println("Goool de " + visitante.getConvocatoria()[tirador].getNombre() + " ("
          + visitante.getNombre() + ")");
    } else {
      System.out.println("Fallo de " + visitante.getConvocatoria()[tirador].getNombre() + " ("
          + visitante.getNombre() + ")");
    }

  }

  // Métodos de aleatoriedad

  private boolean aleatoriedad() {
    return ((int) (Math.random() * 100)) % 2 == 0;
  }

  private boolean tiro(int tiro) {
    return (int) (Math.random() * 100) < tiro && fallo();
  }

  private boolean fallo() {
    return aleatoriedad() && aleatoriedad() && aleatoriedad() && aleatoriedad() && aleatoriedad() && aleatoriedad();
  }

  private boolean isPenalti() {
    return aleatoriedad() && aleatoriedad() && aleatoriedad() && aleatoriedad();
  }

  private boolean parada(Equipo equipo) {

    boolean parada = false;

    if (aleatoriedad()) {
      parada = (int) (Math.random() * 100) <= equipo.getPortero().getParada();
    } else {
      parada = (int) (Math.random() * 100) <= equipo
          .getConvocatoria()[(int) (Math.random() * equipo.getConvocatoria().length)].getParada();
    }

    return parada;

  }

  private boolean penalti(int tiro) {
    return (int) (Math.random() * 100) < (tiro * 2);
  }

  private boolean paradaPenalti(Equipo equipo) {
    return (int) (Math.random() * 100) <= equipo.getPortero().getParada() && aleatoriedad();
  }

  // Métodos de espera

  private void espera() {
    try {
      Thread.sleep(300);
    } catch (InterruptedException exc) {
```

```
      System.out.println("ERROR - Fallo en la espera");
    }
  }

  private void esperaLenta() {
    try {
      Thread.sleep(3000);
    } catch (InterruptedException exc) {
      System.out.println("ERROR - Fallo en la espera");
    }
  }

}
```

Clase Equipo

```
public class Equipo extends Estadio {

  // Atributos del equipo

  private String nombre;
  private Jugador[] convocatoria;

  // Constructores (Overloading)

  public Equipo() {
    super();
    this.nombre = "";
    this.convocatoria = null;
  }

  public Equipo(String nombre, String estadio, Jugador[] convocatoria) {
    super(estadio);
    this.nombre = nombre;
    this.convocatoria = convocatoria;
  }

  // Getters

  public String getNombre() {
    return nombre;
  }

  public Jugador[] getConvocatoria() {
    return convocatoria;
  }

  // Metodo para obtener el jugador con mejor parada (portero)

  public Jugador getPortero() {

    Jugador portero = convocatoria[0];

    for (Jugador jugador : convocatoria) {
      if (jugador.getPosicion().equalsIgnoreCase("POR")) {
        portero = jugador;
      }
    }

    return portero;

  }

  // Setters

  public void setNombre(String nombre) {
    this.nombre = nombre;
```

```
  }

  public void setConvocatoria(Jugador[] convocatoria) {
    this.convocatoria = convocatoria;
  }

  // toString (Overriding)

  @Override
  public String toString() {

    String listaJugadores = "";

    for (Jugador jugador : convocatoria) {
      listaJugadores += "\n" + jugador.toString();
    }

    return "\n" + getNombre() + super.toString() + listaJugadores;

  }

}
```

Clase Estadio

```
public class Estadio {

    private String nombre;

    public Estadio() {
        this.nombre = "";
    }

    public Estadio(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String toString() {
        return " (" + nombre + ")";
    }

}
```

Clase Jugador

```
public class Jugador {

  // Atributos del jugador

  private String nombre;
  private int dorsal;
  private String posicion;
  private int tiro;
  private int parada;
```

```
  // Constructores (Overloading)

  public Jugador() {
    this.nombre = "";
    this.dorsal = 0;
    this.tiro = 0;
    this.parada = 0;
  }

  public Jugador(String nombre, int dorsal, String posicion, int tiro, int parada) {
    this.nombre = nombre;
    this.dorsal = dorsal;
    this.posicion = posicion;
    this.tiro = tiro;
    this.parada = parada;
  }

  // Getters

  public String getNombre() {
    return nombre;
  }

  public String getPosicion() {
    return posicion;
  }

  public int getDorsal() {
    return dorsal;
  }

  public int getTiro() {
    return tiro;
  }

  public int getParada() {
    return parada;
  }

  // Setters

  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public void setDorsal(int dorsal) {
    this.dorsal = dorsal;
  }

  public void setPosicion(String posicion) {
    this.posicion = posicion;
  }

  public void setTiro(int tiro) {
    this.tiro = tiro;
  }

  public void setParada(int parada) {
    this.parada = parada;
  }

  // toString

  public String toString() {
    return getPosicion() + " - " + getNombre() + " (" + getDorsal() + ")";
  }

}
```