Estructurar las necesidades de la práctica

- getAuthors(countryId) → lista de autores (es un ArrayList<Auhtor>)
- getBooks(year, movie) → lista de libros (es un ArrayList<Book>)

```
public class Book {
        private String titulo, identificador, … ;
        public Book (String titulo, String identificador…)  {
                this.titulo  = titulo;
                this.identificador = identificador;
                …
        }

        public String getTitulo () { return this.titulo; }

        …
}
```

Práctica 2

- En el doGet()

```
String fase = request.getParameter(”fase");  // pantalla que se está pidiendo

 switch (fase) {
     case "01":    // se pide la página inicial
                   this.doGetFase01(request, response);
                   break;
     case ”11":    // se pide el listado de years
                   this.doGetFase11(request, response);
                   break;
     …
 }
```

- Todos los ficheros class deben de estar en un directorio classes/p2
  - package p2

Práctica 2

- Por ejemplo, en doGetFase2(request, response)

```
String year = request.getParameter(”pais");

// la clase DataModel contiene métodos estáticos ‘get…’ para calcular resultados
ArrayList<Authores> autores = DataModel.getAuthors (pais);

 response.setCharacterEncoding(‘utf-8’);

FrontEnd.sendHTMLF12(response, year, movies)
```

Práctica 2

- Funciones que obtiene los datos, por ejemplo getAuthors()

```
// Author es una clase que encapsula toda la información de un autor

public ArrayList<Author> getAuthors ( país ) {
    if  (estamos en la fase inicial de datos inventados)
            return new ArrayList<Authors>(Arrays.asList(
                new Author (campo1, campo2,… ), new Author(campo1, campo2), …));
    else {
        // buscar en los DOM la información
        ArrayList<Author> listaAutores = new ArrayList<Author> ()
        bucle estudiando todas los autores encontradas {
            si  cumple  → listaAutores.add(new Author (sus campos))
        }
        ordenar listaAutores;
        return listaAutores;
    }
}
```

Práctica 2

- Ordenaciones
  - Por defecto ArrayList<String> listaCadenas= new ArrayList<String>()

    Collections.sort(listaCadenas); // alfabéticamente

  - Sobre un objeto de creación propia

    ArrayList<Author> listaAutores= new ArrayList<Author>()

    Collections.sort(listaAutores); // por el criterio definido en el objeto Author

```
public class Author implements Comparable< Author > {
   // variables y métodos que pida la clase, y, además
   public int compareTo(Author segudoAutor) {
        compara dos autores (this y segundoAutor) según sus datos
        y devuelve -1, 0, o 1 según su orden
   }
}
```

```
public int compareTo(Author segundoAutor) {
        if (this.name.length() < segundoAutor.name.length()) return 1;
        else …

}
```

CineML

```
<Cine>
    <Pais>
        <NombrePais> pais1 </NombrePais>
        <Pelicula>
            <NombrePeli> pelicula1 </NombrePeli>
            <Guionista nacionalidad="n1">
                <NombreGuio> guio1 </NombreGuio>
            </Guionista >
            … más guionistas
        </Pelicula>
        … más películas
    </Pais>
    … más países
</Cine>
```

Búsqueda

- Búsqueda de los guionistas de una película perteneciente a un país
- Recibe: documento, país y película
- Devuelve: lista de guionistas (objetos *Guionista*)
- Sin duplicidades
- Ordenada

```
class  Guionista {
      String nombre;
      String nacionalidad;
      …
}
```

Búsqueda secuencial de guionistas

```
ArrayList<Guionista>  getGuionistass (Document doc, String pais, String pelicula) {
   ArrayList<Guionista> listaGuionistas = new ArrayList<Guionista>();

   NodeList nlPaises = doc.getElementsByTagName("Pais");

   for (int pa=0; pa < nlPaises.getLength(); pa++) {
      Element elemPais = (Element)nlPaises.item(pa);

      NodeList nlNombresPais = elemPais.getElementsByTagName("NombrePais");
      Element elemNombrePais = (Element)nlNombresPais.item(0);
      String nombrePais = elemNombrePais.getTextContent().trim();

      if (nombrePais.equals(pais)) { // hemos encontrado el país
            NodeList nlPeliculas = elemPais.getElementsByTagName("Pelicula");
```

Búsqueda secuencial de guionistas

```
if (nombrePais.equals(pais)) { // hemos encontrado el país
        NodeList nlPeliculas = elemPais.getElementsByTagName("Pelicula");

        for (int pe=0; pe < nlPeliculas.getLength(); pe++) {
                Element elemPelicula = (Element)nlPeliculas.item(pe);

                NodeList nlNombresPelicula = elemPelicula.getElementsByTagName("NombrePeli");
                Element elemNombrePelicula = (Element)nlNombresPelicula.item(0);
                String nombrePelicula = elemNombrePelicula.getTextContent().trim();

                if (nombrePelicula.equals(pelicula)) { // hemos encontrado la película
                        NodeList nlGuionistas = elemPelicula.getElementsByTagName(”Guionista");
```

Búsqueda secuencial de guionistas

```
            if (nombrePelicula.equals(pelicula)) { // hemos encontrado la película
                NodeList nlGuionistas = elemPelicula.getElementsByTagName(”Guionista");

                for (int g=0; g < nlGuionistas.getLength(); g++) {
                    Element elemGuionista = (Element)nlGuionistas.item(g);

                    NodeList nlNombresGuionista = elemGuionista.getElementsByTagName("NombreGuio");
                    Element elemNombreGuionista = (Element)nlNombresGuionista.item(0);
                    String nombreActorGuionista = elemNombreGuionista.getTextContent().trim();

                    String nacionalidad = elemGuionista.getAttribute("nacionalidad");
                    // evitar duplicidades si necesario
                    Guionista guioObj = new Guionista(nombreGuio, nacionalidad);
                    listaGuionistas.add(guioObj);
                } // final del bucle que recorre guionistas
            } // if nombrePelicula
        } // final del bucle que recorre las películas
    } // if nombrePais
} // final del bucle que recorre los países
// ordenar la lista si necesario
return listaGuionistas;
```