NAME		GRADE	
		1ST - DAM / DAW	
SURNAMES		SUBJECT	ANNOUNCEMENT
		PROGRAMMING	2ND EVALUATION – LAST ATTENT
DNI	DATE	GRADE	
	20/03/2023		

From a CSV file looking like this:

Nombre	Apellidos	DNI	Edad	Mail	Experiencia	ITRole	Departamento
Pedro	Lopez Sanz	98734527F	33	plopez@informatica.es	10	BackEnd	
Marta	Martinez Gonzalez	11142455X	22	mmartinez@informatica.es	2	FrontEnd	
Carlota	Martinez Ramos	67453398D	52	cmartinez@informatica.es	30		Board
Carmen	Garcia Lujan	11422358I	23	cgarcia@informatica.es	2		RRHH
Emma	Hueso Ramos	433829873F	36	ehueso@informatica.es	12	BackEnd	
Lorena	Tarrega Navarro	14323451W	19	ltarrega@informatica.es	1	FullStack	
Ramon	Anton Navarro	22344213D	29	ranton@informatica.es	7	FullStack	
Jorge	Martinez Lopez	12325216V	64	jmartinez@informatica.es	40		RRHH
Sergio	Romero Jimenez	11251266E	31	sromero@informatica.es	10	FrontEnd	
Roberto	Jimenez Navarro	19399794X	20	rjimenez@informatica.es	1	FrontEnd	

Or like this (in both cases, is the same csv file):

```
Nombre;Apellidos;DNI;Edad;Mail;Experiencia;ITRole;Departamento
Pedro;Lopez Sanz;98734527F;33;plopez@informatica.es;10;BackEnd;
Marta;Martinez Gonzalez;11142455X;22;mmartinez@informatica.es;2;FrontEnd;
Carlota;Martinez Ramos;67453398D;52;cmartinez@informatica.es;30;;Board
Carmen;Garcia Lujan;11422358I;23;cgarcia@informatica.es;2;;RRHH
Emma;Hueso Ramos;433829873F;36;ehueso@informatica.es;12;BackEnd;
Lorena;Tarrega Navarro;14323451W;19;ltarrega@informatica.es;1;FullStack;
Ramon;Anton Navarro;22344213D;29;ranton@informatica.es;7;FullStack;
Jorge;Martinez Lopez;12325216V;64;jmartinez@informatica.es;40;;RRHH
Sergio;Romero Jimenez;11251266E;31;sromero@informatica.es;10;FrontEnd;
Roberto;Jimenez Navarro;19399794X;20;rjimenez@informatica.es;1;FrontEnd;
```

You must create **an application to manage IT business companies**. In concrete, some components like **managing the payroll**. To do that:

NAME & SURNAMES

1. **CREATE THE DATA MODEL (4 POINTS)**
   a. **(1p)** Implement an **abstract class** called **Worker** having following elements:
      i. attributes: nombre (String), apellidos (String), DNI (String), edad (int), email (String) and experiencia (int). All of them with a private visibility except experiencia which can be accessed from the subclasses and classes in the same package.
      ii. constructor with all fields.
      iii. an abstract method called getRole having no parameters and returning a String. For IT workers the role will be the category, and in case of non it workers the department they belong to.
      iv. Implement getters for all the attributes.
      v. Implement equals method knowing that a person is equals to another person if their DNIs are the same.
      vi. Implements those methods needed **depending on the interfaces you think you need to implement** or any other method depending on the Collections you will need.

   b. **(0,75p)** Implement a **class** called **ITWorker** which inherits from Worker having:
      i. An attribute called category (Category). This category attribute is an enum called Category.
      ii. Constructor with all the needed attributes
      iii. Implement those methods you think are needed.

   c. **(0.50p)** Implement an **enum class** called **Category** having following values: FULLSTACK, FRONTEND, BACKEND. This enum class will have a String containing the category description as written in the csv file. We need the constructor, a get method for the attribute and a static method called getCategoryFromString. This method will have a String as parameter and returning a Category value. Additionally, this method will throw an Exception (from Exception class) in case the integer parameter is not an appropriate value ("FrontEnd", "BackEnd", "FullStack").

   d. **(0.75p)** Implement a **class** called **NonITWorkers** which inherits from Worker having:
      i. Attributes: a department (Department).
      ii. Constructor with all the needed attributes.
      iii. Implement getters for all the attributes.
      iv. Implement those methods you think are needed.

NAME & SURNAMES

e. **(0.50p)** Implement an **enum class** called **Department** having following values: BOARD, RRHH. This enum class will have a String containing the category description as written in the csv file. We need the constructor, a get method for the attribute and a static method called getDepartamentFromString. This method will have a String as parameter and returning a Department value. Additionally, this method will throw an Exception (from Exception class) in case the integer parameter is not an appropriate value ("Board", "RRHH").

f. **(0.50p)** Implement an **Interface** called **Payable** having three methods which will be coded by all the classes implementing it:
   i. getFullName (returning a String, having no parameters)
   ii. getYearsExperience (returning an integer, having no parameters)
   iii. getRole (returning a String, having no parameters).

2. **(6 POINTS)** Perform the following methods which theoretically will be implemented in a Main class:

   a. **(1p)** A method (cargarDatos) to get a list of Workers from a file name (String). The file, a CVS one, will look like the one in the header of this exam.

   b. **(1p)** A method (getWorkersSorted) to get a sorted by name collection of Payable objects from a list of Persona. This is the **natural sort** for a Worker. Perform any change in the data model classes if needed.

   c. **(1p)** A method (getITbyRole) to get a map of IT Category and a list of IT workers from a listof Personas (Map<Category,List<ITWorker>).

   d. **(1p)** A method (printPayable) to print by console (so, it returns nothing) all the data from a collection of Payable. The result should be the full name, the years of experience and the role. Something like this:

```
[…]
Name: Ramon Anton Navarro YoE: 7 Role: FullStack
Name: Carmen Garcia Lujan YoE: 2 Role: RRHH
Name: Emma Hueso Ramos YoE: 12 Role: BackEnd
[…]
```

NAME & SURNAMES

e. **(1p)** A method (getITWorkersSortedByExperience) to get a descendent sorted list of IT workers by experience from a list of workers. Perform any change in the data model classes if needed.

f. **(1p)** A method (saveAsObject) to save all the Worker objects in a file (called workers.dat). This method has a parameter, a list of workers, and returns nothing. All the potential exceptions are managed in this method. Perform any change in the data model classes if needed.