CONTEXTO

Se desea implementar parte de un sistema relacionado con la gestión de aplicaciones para móviles. Para ello se consideran los siguientes tipos:

App:

- Propiedades:
  - **SO**, de tipo enumerado (ANDROID, IOS, WINDOWS), consultable.
  - **Nombre**, de tipo String, consultable.
  - **Tamaño**, de tipo Double, consultable y modificable.
  - **Fecha de Versión**, de tipo LocalDate, consultable y modificable.
  - **Número de descargas**, de tipo Integer, consultable y modificable.
  - **Precio**, de tipo Double, consultable y modificable.
  - **Desarrollador**, de tipo String, consultable.
  - **Es gratis**, de tipo Boolean, consultable. Será `true` si el precio es 0.
  - **Es novedad**, de tipo Boolean, consultable. Será `true` si la fecha de la versión es reciente (como máximo una semana de antigüedad).
- Constructores: uno que recibe un parámetro por cada una de las propiedades básicas anteriores y en el mismo orden en que están presentadas. También un constructor a partir de String (el formato de la cadena es irrelevante para este examen).
- Criterio de igualdad: según nombre, desarrollador y SO.
- Orden natural: por nombre, en caso de igualdad primero por desarrollador y luego por SO.
- Representación como cadena: nombre, desarrollador, SO, fecha de la versión.
- Restricciones: el número de descargas y el precio deben ser mayor o igual que 0. El tamaño debe ser mayor estrictamente que 0. Si no se cumple una restricción, se lanza la excepción `ExcepcionAppNoValida`.

Opinion:

- Propiedades:
  - **Fecha y hora**, de tipo LocalDateTime, consultable.
  - **Puntuación**, de tipo Integer, consultable.
  - **Texto**, de tipo String, consultable.
- Constructor: recibe un parámetro por cada propiedad y en el mismo orden en que están presentadas.
- Criterio de igualdad: por texto y fecha/hora.
- Orden natural: según la fecha/hora y en caso de coincidencia por texto.
- Representación como cadena: el texto de la opinión junto a la fecha/hora.
- Restricción: la puntuación toma valores entre uno y cinco, ambos inclusive.

AppTienda, subtipo de App**:**

- Propiedades:
  - **Detalles**, de tipo String, consultable y modificable.
  - **Categoría**, de tipo Categoría, tipo enumerado con los siguientes valores: PRODUCTIVIDAD, MULTIMEDIA, JUEGO, NIÑOS y OTROS. Consultable.
  - **Opiniones**, de tipo `SortedSet<Opinion>`, consultable. Las opiniones están ordenadas por el orden natural de Opinion.
  - **Apps Relacionadas**, de tipo `Set<AppTienda>`, consultable y modificable.
  - **Numero de Opiniones**, de tipo Integer, consultable.
- Operaciones:
  - Añade opinión, añade una opinión dada a la propiedad opiniones. Si dicha opinión ya estaba entonces no se añadirá.
  - Borra opinión, borra una opinión dada del conjunto de opiniones.

  (Existen más operaciones que se le pedirán más adelante)
- Constructor: análogo al de App (mismos parámetros de entrada y en el mismo orden), además de un parámetro de tipo String para los detalles, otro de tipo Categoría para la categoría, y otro de tipo `Set<AppTienda>` para las apps relacionadas. El conjunto de opiniones estará vacío al crearse el objeto.
- Criterio de igualdad: análogo al de App.
- Orden natural: análogo al de App.
- Representación como cadena: análogo al de App, seguido de la cadena "Detalles: " y a continuación la propiedad detalles.
- Restricción: los detalles no pueden ser una cadena vacía. Las apps relacionadas deben ser del mismo SO que la app en cuestión.

Tienda:

- Propiedades:
  - **Apps**, de tipo `Set<AppTienda>`, consultable.
  - **Apps por categoría**, de tipo `Map<Categoria, Set<AppTienda>>`, consultable (propiedad derivada).
- Operaciones:
  - Añade app, añade una `AppTienda` dada a la tienda. Si la app ya existe, pero la fecha de la versión es anterior, se sustituirá una app por la otra. Si la app ya existe, pero la fecha de la versión es igual o posterior, se lanzará una excepción `IllegalArgumentException`.
  - Borra app, borra una `AppTienda` dada de la tienda.

  (Existen más operaciones que se le pedirán más adelante)
- Constructor: no recibe parámetros, y crea una tienda vacía, sin apps.
- Criterio de igualdad: dos tiendas son iguales si tienen las mismas apps.
- Representación como cadena: consiste en la representación como cadena de todas las apps de la tienda.

	FUNDAMENTOS DE PROGRAMACIÓN **EXAMEN FINAL. 25 de junio de 2015**	**Curso: 2014/15** **Versión: 1.0.7**

PRIMERA PARTE

EJERCICIO 1: IMPLEMENTACIÓN DE TIPOS (2 puntos)

Escriba, para la clase `AppImpl`, que implementa la interfaz `App`:

a) La cabecera y los atributos de la clase.
b) El constructor con parámetros.
c) Los métodos consultores y modificadores necesarios relacionados con las propiedades número de descargas, es gratis y es novedad.
d) Los métodos `equals` y `compareTo`.

EJERCICIO 2: HERENCIA (2 puntos)

Se pide implementar el tipo `AppTiendaImpl` reutilizando por herencia el tipo `AppImpl`. Para ello, complete **exclusivamente** los siguientes elementos:

a) Defina la cabecera y los atributos necesarios para la clase.
b) Defina el constructor de `AppTiendaImpl`. Suponga ya implementados los métodos privados necesarios para controlar las restricciones del tipo **(no se pide su implementación)**.
c) Defina los métodos necesarios para completar el criterio de igualdad, el orden natural y la representación como cadena siempre y cuando sean **estrictamente necesarios**.

EJERCICIO 3: COLLECTION, LIST, SET, SORTEDSET (2 puntos)

a) Implemente el método `appsComunes(Tienda t, Categoria c)` de la clase `TiendaImpl` que devuelva un conjunto con aquellas apps comunes con la tienda `t` que sean de categoría `c`. **No se permite utilizar bucles ni la clase Stream de Java 8**.
b) Implemente el método `textosComentariosAnteriores(LocalDateTime fecha)` de la clase `AppTiendaImpl` que devuelva un conjunto ordenado alfabéticamente con todos los textos de todas las opiniones anteriores a la fecha y hora dadas. **Sólo puede utilizar un bucle para resolver este ejercicio. NOTA:** Dese cuenta de que el criterio de orden natural del tipo Opinion es por fecha/hora.

EJERCICIO 4: TRATAMIENTOS SECUENCIALES MEDIANTE BUCLES (4 puntos)

Implemente métodos estáticos para resolver los siguientes tratamientos secuenciales. Implemente los tratamientos sin usar la clase `Stream` de Java 8 (es decir, mediante los esquemas basados en bucles estudiados en el curso).

a) Dada una tienda, devolver la app gratis con mayor número de descargas. Si no hay ninguna app gratis, se devolverá `null`.
```
public static AppTienda gratisConMasDescargas(Tienda t)
```
b) Dada una tienda, devolver la media del tamaño de las apps que corran bajo IOS. Si no hubiese ninguna, se devolverá cero.
```
public static Double promedioTamañoIOS(Tienda t)
```
c) Dados un `Set<AppTienda>` y una puntuación `n`, devolver el número de apps con todas las puntuaciones mayores que `n`.
```
public static Integer numeroAppsConPuntuacionMayorN(Set<AppTienda> apps,Integer n)
```

SEGUNDA PARTE

EJERCICIO 5: CONSTRUCTOR A PARTIR DE STRING (1,5 puntos)

Implemente en la clase `OpinionImpl` un constructor que permita construir un objeto a partir de una cadena de caracteres. Ejemplo de formato de la cadena:

> "2015-06-04 18:25# 3# No Funciona. Cada vez que entro siempre sale que se ha producido un error.... Así lleva más de 2 semanas. De pena!!"

Suponga definido un método `void checkPuntuacion(Integer puntuacion)` que comprueba que la puntuación toma valores entre uno y cinco, ambos inclusive. Utilícelo donde proceda. En la representación como cadena pueden aparecer espacios en blanco entre los separadores y los valores de las propiedades. Utilice el patrón "yyyy-MM-dd HH:mm" para *parsear* la fecha y hora.

EJERCICIO 6: FACTORÍAS (1,5 puntos)

Se desea implementar una factoría de objetos de tipo App. Para ello, en una clase de utilidad `Apps`, implemente las siguientes propiedades poblacionales:

a) Número total de objetos App creados mediante la factoría.
b) Lista con todos los objetos App creados mediante la factoría.
c) Precio medio de todos los objetos App creados mediante la factoría.

Utilice los siguientes nombres para los métodos: `a) getNumAppsCreadas`, `b) getAppsCreadas` y `c) getPrecioMedioAppsCreadas`. **Ningún método puede ser resuelto utilizando bucles ni la clase Stream de Java 8**.

En la misma clase `Apps`, implemente dos métodos creacionales para el tipo App: un método creacional copia y un método creacional a partir de String. Llame a ambos métodos `createApp`.

EJERCICIO 7: MAP Y TRATAMIENTOS SECUENCIALES CON STREAM (4,5 puntos)

a) Implemente en la clase `TiendaImpl` el método consultor de las propiedad derivada "Apps por categoría" del tipo `Tienda`. Utilice para ello **un sólo bucle**. **No se permite utilizar la clase Stream de Java 8**.

Implemente los siguientes métodos de la clase `TiendaImpl`, **utilizando exclusivamente la clase Stream de Java 8 (no se permite usar bucles)**:

b) Un método `novedadesPorNumeroDescargas`, que devuelva una lista con las apps que sean novedad, ordenadas por número de descargas descendentemente.
c) Un método `appMasOpiniones`, que devuelva la app de la tienda que ha recibido un mayor número de opiniones. En caso de que ninguna app tenga opiniones, se devolverá `null`.
d) Un método `todasNovedadesBienValoradas`, que compruebe que todas las novedades de la tienda son bien valoradas en todas las opiniones, y devuelva `true` si lo son, o `false`, en caso contrario. Una app se considera bien valorada si su puntuación es igual o superior a 4.
e) Un método `precioJuegos`, que devuelva la suma del importe total recaudado por los juegos descargados de la tienda.

EJERCICIO 8: LENGUAJE C (2,5 puntos)

Dados los siguientes tipos, que se consideran ya implementados:

- **Cadena**: capaz de almacenar un máximo de 255 caracteres.
- **FechaHora**: tipo struct que contiene los campos de tipo entero dia, mes, año, hora y minuto.
- **App**: tipo struct cuya descripción no interesa para este ejercicio.
- **Logico**: enumerado que puede tomar los valores FALSO y CIERTO.

Y dadas las siguientes funciones (**se suponen ya implementadas y puede utilizarlas en los ejercicios que se piden**):

- Logico **igualApp**(App app1, App app2) que devuelve CIERTO si las app1 y app2 son iguales y FALSO en caso contrario.
- void **leeOpinionFichero**(FILE*, POpinion) que lee un registro de Opinion del fichero cuyo puntero se pasa como primer parámetro y devuelve los valores leídos en el registro cuya dirección se indica en el segundo.
- void **muestraApp** (App) que visualiza por la consola los datos de una app.
- void **muestraFechaYHora** (FechaHora) que visualiza por la consola los datos de una fecha y hora.

Responda a las siguientes cuestiones:

a) Defina en lenguaje C:
   - Un tipo **Opinion** (sin tilde en la última o), con las siguientes propiedades:
     - App, de tipo App.
     - Fecha y hora, de tipo FechaHora.
     - Puntuación, de tipo real.
     - Texto, de tipo Cadena.
   - Un tipo **ArrayOpiniones** para almacenar 1000 elementos como máximo.
   - Un puntero **POpinion** al tipo Opinion.

b) Escriba una función **void muestraOpiniones (…)** que recibiendo como parámetro de entrada un array de opiniones, visualice por la consola todos los elementos del array.

c) Escriba una función void **leeOpinionesFichero(…)** que recibiendo como parámetro de entrada el nombre de un fichero, devuelva los registros leídos en un array de opiniones que se pasa también como parámetro (observe que la función es de tipo void).