Práctica 2: **Herencia**

Programación II

Abril 2020 (versión 08/03/2020)

DLSI – Universidad de Alicante

Alicia Garrido Alenda

Normas generales

- El plazo de entrega para esta práctica se abrirá el miércoles 1 de abril a las 9:00 horas y se **cerrará el miércoles 8 de abril a las 23:59 horas**. No se admitirán entregas fuera de plazo.
- Se debe entregar la práctica en un fichero comprimido de la siguiente manera:

  1. Abrir un terminal.
  2. Situarse en el directorio donde se encuentran los ficheros fuente (`.java`) de manera que al ejecutar `ls` se muestren los ficheros que hemos creado para la práctica y ningún directorio.
  3. Ejecutar:

     ```
     tar cfvz  practica2.tgz Tipo.java Objeto.java Persona.java Tienda.java
     Emisor.java Influenciable.java Celestial.java Querube.java Diablo.java
     ```

Normas generales comunes a todas las prácticas

1. La práctica se debe entregar exclusivamente a través del servidor de prácticas del departamento de Lenguajes y Sistemas Informáticos, al que se puede acceder desde la página principal del departamento (`http://www.dlsi.ua.es`, enlace "Entrega de prácticas") o directamente en `http://pracdlsi.dlsi.ua.es`.

   - **Bajo ningún concepto** se admitirán entregas de prácticas por otros medios (correo electrónico, UACloud, etc.).
   - El usuario y contraseña para entregar prácticas es el mismo que se utiliza en UACloud.
   - La práctica se puede entregar varias veces, pero sólo se corregirá la última entrega.

2. El programa debe poder ser compilado sin errores con el compilador de Java existente en la distribución de Linux de los laboratorios de prácticas; si la práctica no se puede compilar su calificación será 0. Se recomienda compilar y probar la práctica con el autocorrector durante su implementación para detectar y corregir errores.

3. La práctica debe ser un trabajo original de la persona que entrega; **en caso de detectarse indicios de copia con una o más prácticas (o compartición de código) la calificación de la práctica será** `0` y se enviará un informe al respecto tanto a la dirección del departamento como de la titulación.

4. Se recomienda que los ficheros fuente estén adecuadamente documentados, con comentarios donde se considere necesario.

5. La primera corrección de la práctica se realizará de forma automática, por lo que es imprescindible respetar estrictamente los formatos de salida que se indican en este enunciado.

6. El cálculo de la nota de la práctica y su influencia en la nota final de la asignatura se detallan en las transparencias de la presentación de la asignatura.

7. Para evitar errores de compilación debido a codificación de caracteres que **descuenta** `0.5` **de la nota de la práctica**, se debe seguir una de estas dos opciones:

   a) Utilizar EXCLUSIVAMENTE caracteres ASCII (el alfabeto inglés) en vuestros ficheros, **incluídos los comentarios**. Es decir, no poner acentos, ni ñ, ni ç, etcétera.

   b) Entrar en el menú de Eclipse `Edit > Set Encoding`, aparecerá una ventana en la que hay que seleccionar `UTF-8` como codificación para los caracteres.

8. El tiempo estimado por el corrector para ejecutar cada prueba debe ser suficiente para que finalice su ejecución correctamente, en otro caso se invoca un proceso que obliga a la finalización de la ejecución de esa prueba y se considera que dicha prueba no funciona correctamente.

Descripción del problema

En un mundo tan material en el que el estado anímico de las personas depende exclusivamente de las pertenencias que posean, el mundo espiritual ha decidido intervenir. Un ser **Celestial** puede influir positiva o negativamente en los adeptos que tenga o en los objetos, dependiendo de su naturaleza, ya que hay dos tipos de seres celestiales. Un **Querube** es un ser celestial bondadoso, que influirá positivamente en las acciones y estado de sus adeptos, mientras un **Diablo** lo hará negativamente.

No todas las personas se dejan influenciar por los seres celestiales, solo un **Influenciable** modificará su comportamiento en función de la naturaleza de su deidad.

También han aparecido un nuevo tipo de objetos que emiten ondas positivas o negativas. Un **Emisor** es un objeto que emite ondas que afectan a su valor e incluso al de los objetos que lo rodean si está almacenado en una tienda. Además un emisor no puede estar almacenado en más de una tienda al mismo tiempo.

Para esta práctica son necesarias las clases **Objeto**, **Persona**, **Tienda** y el enumerado **Tipo** implementados en la práctica anterior. Además es necesario implementar las clases que se definen a continuación, a las que se pueden añadir variables de instancia y/o clase (siempre que sean privadas) y métodos cuando se considere necesario (que pueden ser públicos o privados).

Un **Emisor** es un **Objeto** que además se caracteriza por:

- *ondas*: entero que indica la potencia y naturaleza de las ondas que emite cuyo rango será `[-3,3]` (`int`).
- *lugar*: tienda en la que se ha almacenado el emisor (`Tienda`).

Un **Emisor** puede realizar las mismas acciones que un **Objeto**, pero algunas de ellas las realiza de forma diferente:

- constructor: se le pasa por parámetro un tipo, una cadena y un entero para el *tipo*, *nombre* e *influencia* del objeto. Inicialmente el valor de sus ondas es `1` y no está almacenado en ninguna tienda.
- *getValorIntrinseco*: devuelve el valor emocional original del objeto.
- *calculaValorEmocional*: calcula el valor emocional original del objeto, lo multiplica por sus ondas y lo devuelve.

Además puede realizar las siguientes acciones:

- *getOndas*: devuelve las ondas que emite.
- *setLugar*: se le pasa por parámetro la tienda en la que va a ser almacenado para que actualice su lugar[1]. Si el parámetro es `null` y el emisor estaba previamente almacenado, se actualiza su lugar, puesto que significa que ya no está en las existencias de la tienda. En otro caso se actualiza su lugar si no estaba previamente almacenado. El método devuelve cierto si actualiza su lugar y falso en otro caso.

[1]Es necesario modificar el método *almacena* de *Tienda* para que le pase un mensaje cuando almacena un emisor

⊙ *emana*: el emisor cambia la influencia de los objetos que se encuentran en las posiciones adyacentes a la suya dentro de las existencias de la tienda en la que está almacenado que estén dentro de su radio de acción de sus ondas. Por ejemplo, si la imagen siguiente son las existencias de una tienda, y el emisor ocupa la posición marcada con una "e" y tiene un valor de ondas de -2 o de 2, su radio de acción es 2, por tanto cambia la influencia de los objetos marcados con una "m":

m	m	m	m	o
m	m	m	m	o
m	e	m	m	o
m	m	m	m	o
m	m	m	m	o

El emisor cambia la influencia de un objeto adyacente multiplicando la influencia de dicho objeto por las ondas que emite. El método devuelve el número de objetos que han incrementado su influencia.

⊙ *purifica*: hace que las ondas del emisor sean positivas.

⊙ *envilece*: hace que las ondas del emisor sean negativas.

⊙ *cambiaOndas*: se le pasa por parámetro un entero que se suma a sus ondas. Devuelve el valor actual de sus ondas.

⊙ *getLugar*: devuelve la tienda en la que está almacenado.

Un **Influenciable** es una **Persona** que además se caracteriza por:

⁕ *deidad*: celestial por el que está influído (`Celestial`);

⁕ *pareja*: persona a la que se ha unido sentimentalmente (`Influenciable`);

⁕ *relaciones*: array dinámico de personas con las que ha tenido relación (`ArrayList<Persona>`);

⁕ *familia*: array dinámico de influenciables creados a partir de una relación (`ArrayList<Influenciable>`)

Un **Influenciable** puede realizar las mismas acciones que una **Persona**, pero algunas de forma distinta:

⊙ constructor: se le pasa por parámetro un número real, una cadena y un booleano que se asignan a sus variables de instancia *edad*, *nombre* y *genes* respectivamente de la persona. Inicialmente no tiene deidad ni pareja y se crean los arrays para sus relaciones y familia.

⊙ *encuentra*: se le pasa por parámetro un objeto que ha encontrado. Si el influenciable no tiene deidad o el objeto es un objeto normal, se comporta como una persona y por tanto devuelve cierto si lo guarda y falso en caso contrario. Si tiene deidad y el objeto es un emisor y no tiene poseedor, la deidad influye en el emisor y a continuación lo guarda en sus pertenencias añadiéndolo al principio de sus pertenencias, de manera que el emisor ahora sí tiene poseedor y el método devuelve cierto. En cualquier otro caso, el método devuelve falso.

⊙ ***adquiere***: se le pasa por parámetro una tienda y un tipo de objeto. El influenciable desea adquirir de la tienda el emisor de ese tipo con menor coste. El coste de un emisor para un influenciable es el valor absoluto del valor intrínseco del emisor. Para poder realizar la transacción la tienda tiene que disponer de emisores de ese tipo y el influenciable tiene que poderlo pagar. Para poder pagar el coste del emisor el influenciable debe tener un estado anímico superior o igual al coste del emisor. Si se realiza finalmente la adquisión el influenciable decrementa su estado anímico con el coste del emisor, que añade al principio de sus pertenencias, la tienda elimina de sus existencias dicho emisor e incrementa con el coste del mismo sus ganancias. Si se realiza la adquisición el método devuelve cierto. En cualquier otro caso el método devuelve falso.

Y además puede realizar las siguientes acciones:

⊙ ***empareja***: se le pasa por parámetro un influenciable. Si ninguno tiene pareja, no son familia, ambos tienen deidad del mismo tipo, establecen una relación de pareja, de manera que se eliminan de sus respectivas relaciones si estuvieran presentes en ellas, actualizan sus respectivas parejas, incrementan su estado anímico en `3` unidades y el método devuelve cierto. En cualquier otro caso el método devuelve falso.

⊙ ***relaciona***: se le pasa por parámetro una persona que se añade al final de sus relaciones si no lo estaba previamente y no es su pareja. Se pueden dar los siguientes casos:

1. Si la deidad del influenciable es un diablo, la relación puede ser tóxica para la otra persona, de manera que:
   - si la persona pasada por parámetro es un influenciable que tiene como deidad un diablo, ambos comentan sus fechorías e incrementan su estado anímico con la cantidad de relaciones que tienen respectivamente;
   - si la persona pasada por parámetro es un influenciable que tiene como deidad un querube, se establece una relación tensa que se salda con el decremento en el estado de ánimo de ambos en el número de familiares que tengan respectivamente;
   - si la persona pasada por parámetro no es un influenciable o es un influenciable sin deidad, le robará la mitad de su estado anímico, con el que incrementa el propio;
   - si la persona pasada por parámetro es su pareja, se entregan a la lujuria de manera que, si uno es mujer y otro hombre tienen un hijo, es decir se crea un nuevo influenciable que se tiene que añadir al final de la familia de ambos y que tendrá:
     a) como nombre la concatenación del nombre de sus progenitores separados por un espacio en blanco, primero el del hombre y segundo el de la mujer;
     b) los mismos genes del progenitor con menor valor en su estado anímico. Si son iguales, serán los genes de la pareja;
     c) y para la edad se obtiene el módulo entre la suma de las edades de los progenitores y `10`, y este valor indica la posición de ***`Edades`*** que se debe utilizar en este caso;
     d) la deidad del progenitor con mayor estado anímico. Si tienen el mismo estado anímico, será la deidad de la pareja.

     Si han tenido un hijo se incrementa el estado anímico de ambos en `3.5` unidades, en otro caso se incrementa en `1.5`.
2. Si la deidad del influenciable es un querube, la relación será positiva para la otra persona, de manera que:

- si la persona pasada por parámetro es un influenciable que tiene como deidad un querube, ambos entran en un éxtasis de entusiasmo por sus buenas acciones e incrementan sus estados anímicos con la cantidad de relaciones que tienen como deidad un querube respectivamente, siempre que no sean familiares;
- si la persona pasada por parámetro es un influenciable que tiene como deidad un diablo, le hace ver lo erróneo de su conducta, de manera que el otro influenciable deja de tener como deidad al diablo[2], del cual deja de ser adepto, y ambos incrementan su estado anímico en `1.0` unidad;
- si la persona pasada por parámetro no es un influenciable o es un influenciable sin deidad, le incrementa su estado anímico en `1.0` unidad, que decrementa del suyo propio;
- si la persona pasada por parámetro es su pareja, fortalecen su relación de manera que, si uno es mujer y otro hombre tienen un hijo, es decir se crea un nuevo influenciable que se tiene que añadir al final de la familia de ambos y que tendrá:
  a) como nombre la concatenación del nombre de sus progenitores separados por un guión ("`-`"), primero el de la mujer y segundo el del hombre;
  b) para la edad se se suman las edades de los progenitores, y se obtiene el módulo entre esta cantidad y `10`, y este valor indica la posición de ***`Edades`*** que se debe utilizar en este caso.
  c) los mismos genes del progenitor con mayor estado anímico. En caso de tener el mismo estado anímico serán los genes del propio influenciable;
  d) la deidad del progenitor con menor estado anímico, en caso de tener el mismo estado anímico será la deidad del propio influenciable.

  Si han tenido un hijo se incrementa el estado anímico de ambos en `4.0` unidades, en otro caso se incrementa en `2.0` unidades.

3. Si el influenciable no tiene deidad ambos incrementan su estado anímico en `0.5` unidades.

El método devuelve la diferencia entre el estado anímico actual del influenciable y su estado anímico anterior a la relación.

⊙ ***`intercambio`***: se le pasa por parámetro una persona, con la que puede intercambiar un objeto, y dos tipos de objetos. La persona pasada por parámetro no puede ser el propio influenciable, y los tipos pasados por parámetro tienen que ser distintos entre sí. Entonces si el influenciable tiene un objeto entre sus pertenencias cuyo tipo coincida con el primero, y la persona pasada por parámetro tiene a su vez entre las suyas un objeto cuyo tipo coincida con el segundo tipo, se intercambian dichos objetos[3], y el método devuelve cierto (si tienen más de un objeto de ese tipo, intercambian el primero que encuentran de ese tipo). En cualquier otro caso, devuelve falso.

⊙ ***`getFamilia`***: devuelve la variable ***`familia`***.

⊙ ***`getRelaciones`***: devuelve la variable ***`relaciones`***.

⊙ ***`getDeidad`***: devuelve la variable ***`deidad`***.

[2]No tendrá ninguna.

[3]Es decir, los objetos intercambiados ocuparán la misma posición que ocupaba el objeto por el cual se cambia, y se actualizan los nombres de sus poseedores.

⊙ *setDeidad*: se le pasa por parámetro un celestial que se convierte en su *deidad*.

⊙ *getPareja*: devuelve la variable *pareja*.

Un **Celestial** se caracteriza por:

⁕ *nombre*: cadena con el nombre por el que es reconocido por sus adeptos (`String`);

⁕ *aura*: poder celestial que posee (`double`);

⁕ *adeptos*: array dinámico de influenciables con los que ha tenido relación (`ArrayList<Influenciable>`);

Un **Celestial** no existe como tal, siempre será o bien un **Querube** o bien un **Diablo**.

Un **Celestial** puede realizar las siguientes acciones:

⊙ constructor: se le pasa por parámetro una cadena para asignar a su nombre. Si la cadena es vacía o `null` adoptará el nombre de "Ser superior". Inicialmente su aura es `0` y crea el array para sus adeptos.

⊙ *enfrentamiento*: se le pasa por parámetro un celestial y un entero. Si son contrarios, es decir, uno es diablo y el otro querube, se produce un enfrentamiento. En dicho enfrentamiento cada celestial hace acopio de todo el aura que puede; para ello recorre a sus adeptos cogiendo `2.0` unidades de su estado anímico, que les resta y suma a su aura. Además los querubes incrementan su aura con el número de objetos influenciados, dejándolo a `0`. El celestial con mayor aura es el vencedor de dicho enfrentamiento, y como resultado le quita tantos adeptos al contrario como indica el entero pasado por parámetro, si es mayor que `0`, de los últimos que haya añadido y los hace propios, añadiéndolos al final de sus propios adeptos. Si el entero pasado por parámetro es menor o igual a `0` o se produce un empate, no se produce ningún cambio en los adeptos de ninguno de ellos. Tras el enfrentamiento el aura de ambos celestiales se decrementa en el número de adeptos que tengan cada uno de ellos. El método devuelve el nombre del celestial vencedor si lo hay, y en caso de empate devuelve una cadena con el nombre de ambos celestiales concatenados, separados por un `-`, primero el del celestial pasado por parámetro y segundo el propio nombre. Si no se produce enfrentamiento porque ambos celestiales son del mismo tipo, el método devuelve una cadena con el nombre de ambos celestiales concatenados, separados por un `+`, primero el del celestial pasado por parámetro y segundo el propio nombre. El método devuelve `null` por defecto si no se produce ninguna de estas circunstancias.

⊙ *getNombre*: devuelve el nombre del celestial.

⊙ *getAura*: devuelve el aura del celestial.

⊙ *getAdeptos*: devuelve los adeptos del celestial.

Además tiene las siguientes acciones que no puede realizar, ya que se especifican según el tipo de celestial que sea:

⊙ *influye*: se le pasa por parámetro un objeto. Dependiendo de la naturaleza del celestial influirá en el objeto de una manera o de otra.

⊙ *`influye`*: se le pasa por parámetro una persona. Dependiendo de la naturaleza del celestial influirá en la persona de una manera o de otra. El método devolverá el estado anímico de la persona tras la influencia ejercida.

Un **Querube** es un **Celestial** que además se caracteriza por:

⁜ *`influenciados`*: entero que indica el número de objetos que han sido influenciados por el querube (`int`);

Y especifica como realizar las acciones:

⊙ constructor: se le pasa por parámetro una cadena para el nombre del celestial e inicialmente el número de objetos influenciados es `0`.

⊙ *`influye`*: se le pasa por parámetro un objeto. Si se trata de un emisor lo purifica para que afecte positivamente a todo lo que le rodea e incrementa sus ondas en `1` unidad. Si no es un emisor modifica su influencia para que tenga el valor actual pero positiva. Si ya era positiva, la incrementa en `2` unidades. En cualquier caso suma a su aura el valor emocional del objeto pasado por parámetro. El método devuelve el valor emocional del objeto después de la influencia ejercida e incrementa en `1` los objetos influenciados. Por defecto el método devuelve `0`.

⊙ *`influye`*: se le pasa por parámetro una persona. Si es solo una persona normal influye en todas sus pertenencias y actualiza su estado, mientras que si se trata de un influenciable que no tiene deidad, se convierte en su deidad y lo añade al final de sus adeptos. Si se trata de un influenciable que ya tiene deidad, incrementa su estado anímico con una décima parte de su aura de manera que la decrementa en dicha cantidad. El método devuelve el estado anímico de la persona después de la influencia ejercida. Por defecto el método devuelve `0`.

Además puede realizar las acciones:

⊙ *`getInfluenciados`*: devuelve el número de objetos influenciados hasta el momento.

⊙ *`obsequia`*: se le pasa por parámetro una tienda. Recorre a sus adeptos, obteniendo los familiares de cada adepto, de manera que para el primer familiar de cada adepto el querube elimina de la tienda el primer objeto que tenga almacenado en ese momento, y se lo da al familiar para que lo encuentre. El método devuelve un array dinámico con los nombres de todos los familiares que han encontrado un regalo.

Un **Diablo** es un **Celestial** sin ninguna característica adicional pero que especifica como realizar las acciones:

⊙ constructor: se le pasa por parámetro una cadena para el nombre del celestial.

⊙ *`influye`*: se le pasa por parámetro un objeto. Si se trata de un emisor lo envilece para que afecte negativamente a todo lo que le rodea. Si no es un emisor suma a su aura el valor emocional de dicho objeto y a continuación modifica su influencia para que tenga el valor actual pero negativa. Si ya era negativa, la decrementa en `1` unidad más. El método devuelve el valor emocional del objeto después de la influencia ejercida. Por defecto el método devuelve `0`.

⊙ *`influye`*: se le pasa por parámetro una persona. Si es solo una persona normal influye en todas sus pertenencias, mientras que si se trata de un influenciable que no tiene deidad, se convierte en su deidad y lo añade al principio de sus adeptos. Si se trata de un influenciable que ya tiene deidad, le roba la cuarta parte de su estado anímico con el que incrementa su aura, y si además tiene pareja hace que rompan su relación de pareja. El método devuelve el estado anímico de la persona después de la influencia ejercida. Por defecto el método devuelve `0`.

Además puede realizar las acciones:

⊙ *`diaboliza`*: consulta la deidad de los familiares de todos sus adeptos, de manera que si alguno tiene otra deidad, el diablo se convierte en su deidad actual. El método devuelve un array dinámico con los nombres de todos los familiares que ha convertido en sus adeptos.

Restricciones en la implementación

⊛ Todas las variables de instancia o clase de las clases deben ser privadas (no accesibles desde cualquier otra clase).

⊛ Algunos métodos deben ser públicos y tener una *`signatura`* concreta:

- En **Emisor**
  - `public Emisor(Tipo,String,int)`
  - `public double getValorIntrinseco()`
  - `public ArrayList<Fruto> recolecta()`
  - `public double calculaValorEmocional()`
  - `public int getOndas()`
  - `public boolean setLugar(Tienda)`
  - `public int emana()`
  - `public void purifica()`
  - `public void envilece()`
  - `public int cambiaOndas(int)`
  - `public Tienda getLugar()`
- En **Influenciable**
  - `public Influenciable(double,String,boolean)`
  - `public boolean encuentra(Objeto)`
  - `public boolean adquiere(Tienda,Tipo)`
  - `public boolean empareja(Influenciable)`
  - `public double relaciona(Persona)`
  - `public boolean intercambio(Persona,Tipo,Tipo)`

  - `public ArrayList<Influenciable> getFamilia()`
  - `public ArrayList<Persona> getRelaciones()`
  - `public Celestial getDeidad()`
  - `public void setDeidad(Celestial)`
  - `public Influenciable getPareja()`

- En **Celestial (abstracta)**
  - `public Celestial(String)`
  - `public String enfrentamiento(Celestial,int)`
  - `public String getNombre()`
  - `public double getAura()`
  - `public ArrayList<Influenciable> getAdeptos()`
  - `public abstract double influye(Objeto)`
  - `public abstract double influye(Persona)`

- En **Querube**
  - `public Querube(String)`
  - `public double influye(Objeto)`
  - `public double influye(Persona)`
  - `public int getInfluenciados()`
  - `public ArrayList<String> obsequia(Tienda)`

- En **Diablo**
  - `public Diablo(String)`
  - `public double influye(Objeto)`
  - `public double influye(Persona)`
  - `public ArrayList<String> diaboliza()`

⊛ Ninguno de los ficheros entregados en esta práctica debe contener un método `public static void main(String[] args)`.

⊛ Todas las variables de clase e instancia deben ser privadas. En otro caso se restará un `0.5` de la nota total de la práctica.

Probar la práctica

- En UACloud se publicará un corrector de la práctica con un conjunto mínimo de pruebas (se recomienda realizar pruebas más exhaustivas de la práctica).

- Podéis enviar vuestras pruebas (fichero con extensión `java`, con un método `main` y sin errores de compilación) a los profesores de la asignatura mediante tutorías de UACloud, para obtener la salida correcta a esa prueba **a partir del 23 de marzo**. En ningún caso se modificará/corregirá el código de las pruebas. Los profesores contestarán a vuestra tutoría adjuntando la salida de vuestro `main`, si no da errores.

- El corrector viene en un archivo comprimido llamado `correctorP2.tgz`. Para descomprimirlo se debe copiar este archivo donde queramos realizar las pruebas de nuestra práctica y ejecutar:

  ```
  tar xfvz correctorP2.tgz
  ```

  De esta manera se extraerán de este archivo:

  - El fichero `corrige.sh`: *shell-script* que tenéis que ejecutar.
  - El directorio `practica2-prueba`: dentro de este directorio están los ficheros con extensión `.java`, programas en Java con un método `main` que realizan una serie de pruebas sobre la práctica, y los ficheros con el mismo nombre pero con extensión `.txt` con la salida correcta para la prueba correspondiente.

- Una vez que tenemos esto, debemos copiar nuestros ficheros fuente (sólo aquellos con extensión `.java`) al mismo directorio donde está el fichero `corrige.sh`.

- Sólo nos queda ejecutar el *shell-script*. Primero debemos darle permisos de ejecución si no los tiene. Para ello ejecutar:

  ```
  chmod +x corrige.sh
  corrige.sh
  ```

- Una vez se ejecuta `corrige.sh` los ficheros que se generarán son:

  - `errores.compilacion`: este fichero sólo se genera si el corrector emite por pantalla el mensaje "`Error de compilacion: 0`" y contiene los errores de compilación resultado de compilar los fuentes de una práctica particular. Para consultar el contenido de este fichero se puede abrir con cualquier editor de textos (gedit, kate, etc.).
  - Fichero con extensión `.tmp.err`: este fichero debe estar vacío por regla general. Sólo contendrá información si el corrector emite el mensaje "`Prueba p01: Error de ejecucion`", por ejemplo para la prueba `p01`, y contendrá los errores de ejecución producidos al ejecutar el fuente `p01` con los ficheros de una práctica particular.
  - Fichero con extensión `.tmp`: fichero de texto con la salida generada por pantalla al ejecutar el fuente correspondiente, por ejemplo `p01.tmp` contendrá la salida generada al ejecutar el fuente `p01` con los ficheros de una práctica particular.

  Si en la prueba no sale el mensaje "`Prueba p01: Ok`", por ejemplo para la prueba `p01`, o algún otro de los comentados anteriormente, significa que hay diferencias en las salidas para esa prueba, por tanto se debe comprobar que diferencias puede haber entre los ficheros `p01.txt` y `p01.tmp`. Para ello ejecutar en línea de comando, dentro del directorio `practica1-prueba`, la orden: `diff -w p01.txt p01.tmp`