Examen de Prácticas de Programación
Ingeniería Informática

Septiembre 2008

1) Conversión entre tipos numéricos

1.a) (0.25 puntos) Indicar si la asignación que aparece en el siguiente código Java es correcta o no y corregirla en caso de que no lo sea. Indicar cuál sería el valor almacenado en la variable `d` como resultado de la asignación:

```
double d; int i=2; char c='a';
d = i + c;  // ¿asignación correcta?
```

1.b) (0.25 puntos) Proceder de igual manera para el siguiente código, indicando cual sería el valor almacenado en la variable `i`:

```
double d=3.72; int i; int j=10;
i = j + d + '\"';   // ¿asignación correcta?
```

1.c) (0.5 puntos) La asignación que aparece a continuación no es correcta. Indicar como corregirla para que el valor almacenado en `i` sea lo más cercano posible al resultado matemático de la expresión (`d*j = 37.2`). Indicar cuál sería el valor almacenado en `i`.

```
double d=3.72; int i; int j=10;
i = d * j;   // asignación incorrecta
```

2) (2.25 puntos) Se dispone de la clase `Compuerta` ya realizada que tiene los métodos mostrados a continuación (la clase tiene más métodos, pero no son relevantes para el problema planteado):

abre

```
public void abre(int gradoApertura) throws Averiada
```

Abre la compuerta el grado de apertura indicado

Throws:

`Averiada` si se detecta que el mecanismo de la compuerta está averiado

gradoApertura

```
public int gradoApertura()
```

Retorna el grado de apertura actual de la compuerta

Returns:

grado de apertura

Se ha detectado un malfuncionamiento intermitente del mecanismo de apertura de las compuertas, de forma que puede ocurrir que el método `abre` lance la excepción `Averiada`

en una llamada y abra la compuerta (sin lanzar la excepción) en la siguiente invocación. También se ha observado que en ocasiones la apertura lograda por alguna compuerta (leída con `gradoApertura`) no es la solicitada en el método `abre`.

Escribir un nuevo método de la clase `Compuerta` que permita corregir en lo posible esta situación. El método recibirá como parámetros el grado de apertura y el número de intentos de apertura (llamadas al método `abre`) que se desea realizar y finalizará al detectar que se ha alcanzado el grado de apertura solicitado. Deberá lanzar la excepción `Averiada` si en todos los intentos de apertura se ha lanzado la excepción `Averiada`. Si al menos uno de los intentos no lanzó la excepción `Averiada`, pero la apertura lograda después de todos los intentos no es la solicitada, deberá lanzar la excepción `AperturaIncorrecta`.

3) (2.25 puntos) Se desea leer un fichero con el siguiente formato:

```
Coche  1234BBB 2005  2006 2007
Coche 5678AAA 2002 2005 2008
Coche 3333CCC 2006
Coche  S1234CC 1995  1999 2002  2006 2008
...
```

Cada una de las líneas comienza con la palabra "Coche", a continuación aparece la matrícula del coche, seguido de un número que es el año de compra y de cero o más números que corresponden a los años en que el vehículo ha sido reparado. Las palabras y números que componen cada una de las líneas se encuentran separadas por uno o más espacios.

Se desea implementar el método:

```
public static LinkedList<Coche> leeFichero(String nomFich)
                     throws FileNotFoundException
```

que permite realizar la lectura del fichero anteriormente descrito. El fichero puede contener errores de formato. Cuando se encuentre una línea con un error de formato se deberá ignorar y continuar procesando los datos de los demás coches existentes en el fichero.

El constructor de la clase `Coche` es:

```
public Coche(String matrícula, int añoCompra,
       LinkedList<Integer> reparaciones)
```

4) (4.5 puntos) Completar el diseño e implementación de la aplicación (únicamente de las partes solicitadas) que verifica los siguientes D-Requerimientos:

Se desea realizar una aplicación que permita a un periodista deportivo llevar las estadísticas de los jugadores de un equipo de fútbol para posteriormente poder valorar su actuación en el partido. Cada jugador se identifica por su nombre y su número de dorsal.

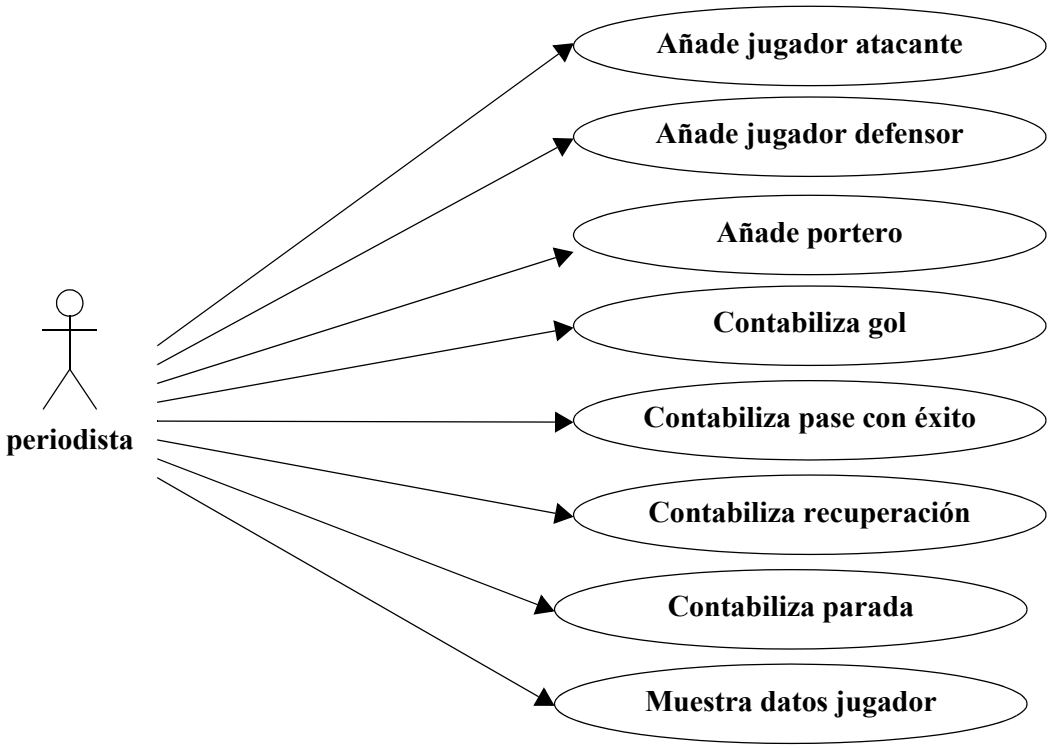
Los jugadores se han dividido en tres categorías: atacantes, defensores y porteros. Para todos los jugadores (incluidos los porteros) se desea contabilizar el número de goles marcados, además

- en el caso de los jugadores de campo (atacantes y defensores) se contabilizaran también los pases realizados con éxito y el número de balones recuperados al contrario.
- en el caso de los porteros, se contabilizará también el número de paradas realizadas.

La valoración de cada jugador se obtendrá según las siguientes fórmulas:

- Atacantes: `goles * 30 + pases * 1 + recuperaciones * 3`
- Defensores: `goles * 20 + pases * 1 + recuperaciones * 4`
- Porteros: `goles * 10 + paradas * 5`

Casos de uso:



A continuación se procede a describir los casos de uso. No se entra en detalles de la interacción entre el periodista y la aplicación (punto 1 de cada caso de uso), puesto que no va a ser tarea del alumno desarrollar esa parte.

Caso de uso "Añade jugador atacante":

1. El periodista elige la opción "Añade jugador atacante" e introduce el nombre y el número del jugador.
2. La aplicación añade el jugador a la lista de jugadores en el equipo (no deberá permitir añadir dos jugadores con el mismo número).

Los casos de uso "Añade jugador defensor" y "Añade portero" son iguales al anterior.

Caso de uso "Contabiliza gol":

1. El periodista elige la opción "Contabiliza gol" e introduce el número del jugador.
2. La aplicación añade el gol al jugador (deberá detectarse el error de que el número no corresponda a ningún jugador)

Caso de uso "Contabiliza pase con éxito":

1. El periodista elige la opción "Contabiliza pase con éxito" e introduce el número del jugador.
2. La aplicación añade el pase al jugador (deberán detectarse los errores de que el número no corresponda a ningún jugador o que el jugador con ese número sea un portero)

El caso de uso "Contabiliza recuperación" es igual al anterior":

Caso de uso "Contabiliza parada":

1. El periodista elige la opción "Contabiliza parada" e introduce el número del jugador.
2. La aplicación añade la parada al jugador (deberán detectarse los errores de que el número no corresponda a ningún jugador o que el jugador con ese número sea un jugador de campo)

Caso de uso "Muestra datos jugador":

1. El periodista elige la opción "Muestra datos jugador" e introduce el número del jugador.
2. La aplicación muestra por consola los datos y la valoración del jugador (deberá detectarse el error de que el número no corresponda a ningún jugador). Dependiendo del tipo de jugador, en la consola deberá aparecer una línea similar a una de las mostradas a continuación:
   - Atacante:
     `8 Lolo(Atacante) goles:1 pases:9 recuperados:3 Valoración:48`
   - Defensor:
     `5 (Defensor) goles:0 pases:7 recuperaciones:9  Valoración:43`
   - Portero:
     `1 Pepe(Portero) goles:0 paradas:12 Valoración:60`

La aplicación contará con un programa principal basado en menú que permitirá al periodista interactuar con la aplicación (dicho programa principal se supone encargado a otro programador, por lo que *no deberá ser realizado por el alumno*).

El alumno deberá desarrollar las clases que permitan gestionar las estadísticas de los jugadores y calcular su valoración.

Se pide:

- diseño arquitectónico de la parte de la aplicación encargada al alumno
- código de las clases correspondientes a la parte de la aplicación encargada al alumno

(Se valorará positivamente la utilización correcta de la herencia en el diseño e implementación de las clases)