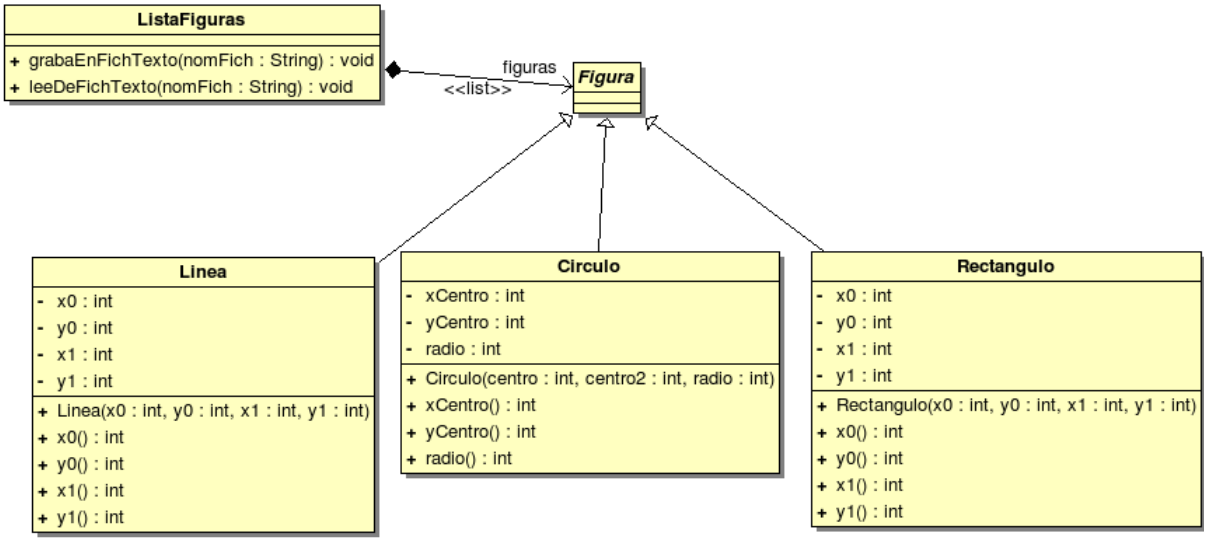
Examen de Prácticas de Programación
Ingeniería Informática

Junio 2010

1) (2 puntos) Todas las clases que aparecen en el diagrama mostrado a continuación están ya implementadas salvo los métodos `grabaEnFichTexto` y `leeDeFichTexto` de la clase `ListaFiguras`.



Las clases podrían tener otros métodos y atributos que no se muestran por no ser relevantes para el problema propuesto.

Implementa los métodos `grabaEnFichTexto` y `leeDeFichTexto` de la clase `ListaFiguras` sabiendo que el atributo `figuras` se ha implementado utilizando un `LinkedList`:

- `grabaEnFichTexto`: graba en el fichero de texto `nomFich` las figuras contenidas en la lista `figuras` del objeto actual.
- `leeDeFichTexto`: reemplaza las figuras contenidas en la lista `figuras` del objeto actual por las contenidas en el fichero de texto `nomFich`.

El formato del fichero deberá ser elegido por el alumno. En el método `leeDeFichTexto` no hay que realizar ningún tratamiento especial para los posibles errores de formato, ya que se considera que el fichero habrá sido creado con el método `grabaEnFichTexto` y que, por lo tanto, no tendrá ningún error.

2) (1.5 puntos) Se dispone de la siguiente clase parcialmente implementada:

```
public class BuscaMCD {
  private static class PrimosEntreSi extends Exception {}
  /**
   * Busca el máximo común divisor de los números a y b
   * @param a uno de los números del que se busca su mcd
   * @param b otro de los números del que se busca su mcd
   * @return máximo común divisor de a y b
   * @throws PrimosEntreSi lanzada si los números son primos entre
   * sí (no existe ningún número aparte del 1 que divida a ambos
   * números)
   */
  private static int mcd(int a, int b) throws PrimosEntreSi {
    // código no relevante para el problema planteado
  }

  /**
   * Calcula el mcd de dos números. Permite al usuario introducir
   * datos hasta que estos sean valores numéricos correctos y no
   * sean primos entre sí
   */
  public static void main(String[] args) {
    // TODO: hacer por el alumno
  }
}
```

Se pide escribir el código del método `main` de forma que pida al usuario dos números (utilizando una ventana de lectura del paquete fundamentos) y muestre en la consola el máximo común divisor de ambos números.

El programa deberá permitir al usuario reintroducir los datos hasta que sean correctos (representen valores numéricos enteros válidos) y correspondan a dos números que no sean primos entre sí.

Un ejemplo de ejecución podría ser:

1. La aplicación muestra la ventana para introducir los datos
2. El usuario introduce los valores: `a=15` y `b=7p`
3. La aplicación muestra un mensaje informando al usuario de que uno de los datos no tiene el formato correcto
4. La aplicación vuelve a mostrar la ventana para introducir los datos
5. El usuario introduce los valores: `a=15` y `b=7`
6. La aplicación muestra un mensaje informando al usuario de que los números introducidos son primos entre sí
7. La aplicación vuelve a mostrar la ventana para introducir los datos
8. El usuario introduce los valores: `a=15` y `b=10`
9. La aplicación muestra por consola el mensaje "El mcd de 15 y 10 es 5" y finaliza la ejecución

3) (1.5 puntos) Responde a las siguientes preguntas relativas al código mostrado a continuación:

- Hay dos líneas que producen un error de compilación: indica cuales son y como corregirlas.
- Con las líneas erróneas corregidas, indica la salida por consola que produciría la ejecución del programa.
- También con las líneas erróneas corregidas, indica los objetos que se crean y en qué línea se produce la creación, así como aquellas líneas en las que un objeto se convierte en basura.

```
import java.util.Arrays;

public class RefsObjs {

  public static void main(String[] args) {
    Object[] cosas = {"uno", new Integer(2),
                      "tres", new StringBuilder("cuatro")};

    System.out.println("Cosas 1:" + Arrays.toString(cosas));

    String str = cosas[0];

    System.out.println("Cosas 2:" + Arrays.toString(cosas));

    str = "HOLA";
    cosas[2] = "TRES";
    cosas[3].append("s");

    System.out.println("Cosas 3:" + Arrays.toString(cosas));
  }
}
```

4) (5 puntos) Completa el diseño e implementación de la aplicación (únicamente de las partes solicitadas) que verifica los siguientes requisitos:

Se desea gestionar la venta de entradas para un espectáculo en un teatro. El patio de butacas del teatro se divide en varias zonas, cada una identificada por su nombre. Los datos de las zonas son los mostrados en la siguiente tabla:

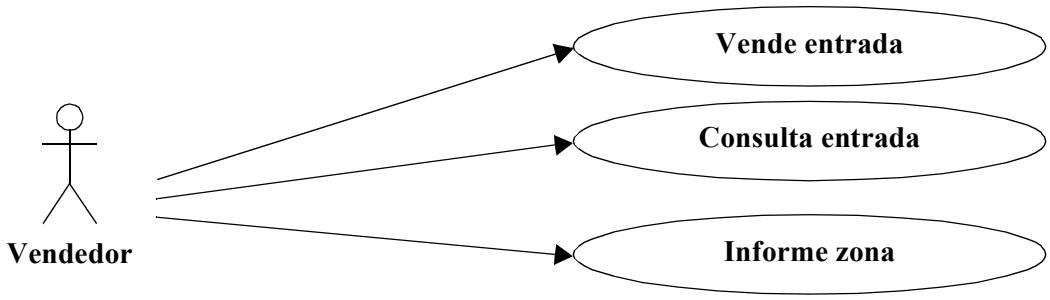
Nombre Zona	Número de localidades	Precio Normal	Precio Abonado
Principal	200	25€	17.5€
Palco	40	70€	40€
Central	400	20€	14€
Lateral	100	15.5€	10€

Para realizar la compra de una entrada, un espectador debe indicar la zona que desea y presentar al vendedor el documento que justifique que tiene algún tipo de descuento (estudiante, pensionista o abonado). El vendedor sacará la entrada del tipo apropiado en la zona indicada. En el momento de la compra se asignará a la entrada un identificador (un número entero) que permitirá la identificación de la entrada en todas las operaciones que posteriormente se desee realizar con ella.

Una entrada tiene como datos asociados su identificador, la zona a la que pertenece y el nombre del comprador. Los precios de las entradas dependen de la zona y del tipo de entrada según lo explicado a continuación:

- Entradas normales: su precio es el precio normal de la zona elegida sin ningún tipo de descuento.
- Entradas reducidas (para estudiantes o pensionistas): su precio tiene una rebaja del 15% sobre el precio normal de la zona elegida.
- Entradas de abonado: su precio es el precio para abonados de la zona elegida.

La interacción entre el vendedor y la aplicación es la descrita en los siguientes casos de uso:



A continuación se procede a describir los casos de uso. No se entra en detalles de la interacción entre el vendedor 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 "Vende entrada":

1. El vendedor elige la opción "vende entrada" e introduce la zona deseada, el nombre del espectador y el tipo (normal, abonado o beneficiario de entrada reducida).
2. Si la zona elegida no está completa, la aplicación genera una nueva entrada con los datos facilitados.

   - Si no existe ninguna zona con ese nombre, se notifica y finaliza el caso de uso sin generar la entrada.

   - Si la zona elegida está completa lo notifica y finaliza el caso de uso sin generar la entrada.
3. La aplicación muestra el identificador y el precio de la entrada.

Caso de uso "Consulta entrada":

1. El vendedor elige la opción "consulta entrada" e introduce el identificador de la entrada.
2. La aplicación muestra los datos de la entrada: nombre del espectador, precio y nombre de la zona

- Si no existe ninguna entrada con ese identificador, lo notifica y finaliza el caso de uso

Caso de uso "Informe zona":

1. El vendedor elige la opción "informe zona" e introduce el nombre de la zona
2. La aplicación muestra el número de localidades vendidas en la zona y la recaudación correspondiente a esas localidades.
   - Si no existe ninguna zona con ese nombre, se notifica y finaliza el caso de uso

La aplicación contaría con un programa principal basado en menú. Para cada caso de uso, ese programa principal se encargaría de gestionar las ventanas que permiten al vendedor introducir los datos solicitados, llamaría al método apropiado de las clases desarrolladas por el alumno y mostraría los datos y/o errores correspondientes a las operaciones realizadas. (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 proporcionan las operaciones que permitirían al programa principal implementar los casos de uso descritos con anterioridad (dichas clases deben gestionar las entradas vendidas, el aforo libre en las distintas zonas, etc.).

Se pide:

- Diagrama de clases (detallado) de la parte de la aplicación encargada al alumno.
- Código de las clases del diagrama que sean el origen de una relación de agregación.
- Código de las clases encargadas de la gestión de las zonas del patio de butacas.
- Código de los constructores de todas las clases involucradas en alguna jerarquía de herencia.
- Código de todos los métodos redefinidos en alguna jerarquía de herencia. Tanto el método de la superclase como el de la(s) subclase(s) (indicar claramente a qué clase corresponde cada método).

(En el código no es necesario incluir los métodos observadores y cambiadores sencillos. Para la evaluación se entenderá que existen todos los métodos de este tipo que aparezcan en el diagrama de clases).