Examen de Prácticas de Programación
Ingeniería Informática

Septiembre 2009

1) Cuestiones

1.a) (1.5 puntos) Se dispone de la siguiente jerarquía de clases:

```
public class Abuelo {
  String s;

  public Abuelo() {
    s = "Sin nombre";
  }

  public Abuelo(String s) {
    this.s = s;
  }

  public String toString() {
    return " s:" + s;
  }
}


public class Padre extends Abuelo {
  public Padre() {
  }

  public Padre(String s) {
    super(s);
  }

  public String toString(int i) {
    return "padre" + super.toString();
  }
}


public class Hijo extends Padre{
  private int i;

  public Hijo(int i) {
    this.i = i;
  }

  public String toString() {
    return super.toString() + ",i:" + i;
  }
}
```

Indicar la salida por consola que se produciría si se ejecuta el siguiente programa:

```
public class Herencia {

  public static void main(String[] args) {
    Hijo h = new Hijo(1);
    Padre p = new Padre("padre");
    Abuelo a1 = new Abuelo("abuelo");
    Abuelo a2 = new Hijo(2);

    System.out.println("h:" + h);
    System.out.println("p:" + p);
    System.out.println("a1:" + a1);
    System.out.println("a2:" + a2);
    System.out.println("h:" + (Abuelo)h);
  }

}
```

1.b) (1.5 puntos) Sobre el código mostrado a continuación, responde a las siguientes cuestiones:

- Indica la línea o líneas que producirían un error de compilación.
- Con las líneas erróneas eliminadas, indica la salida por consola que produciría su ejecución.
- También con las líneas erróneas eliminadas, indica las líneas en las que se produce creación de objetos, así como aquellas líneas en las que un objeto se convierte en basura.

```
import java.util.Arrays;
public class Tipos {
  public static void main(String[] args) {
    Double d1 = 4.5;
    Double d2 = 4.5;

    if (d1.equals(d2)) System.out.println(d1 + " equals " + d2);

    if (d1==d2) System.out.println(d1 + " == " + d2);

    Double[] ds = new Double[2];

    ds[0] = new Double(3.1);
    ds[1] = d2;
    System.out.println("ds:" + Arrays.toString(ds));

    d1 = 2.4;
    d2 = d1;
    System.out.println("ds:" + Arrays.toString(ds));

    char c='a';
    c = c + d2 + d1;
    c = (char)(c + d2 + d1);
    c = c + (char)d2 + d1;
    System.out.println("c:" + c);
  }
}
```

2) (2.75 puntos) Se dispone de la clase vector que permite almacenar vectores de números reales:

```
public class Vector {
  private double[] componentes;

  ... otros métodos no relevantes para el problema

}
```

Se desea añadir a la clase dos métodos que permitan grabar (y recuperar) objetos de la clase `Vector` utilizando *ficheros binarios de tipos primitivos*.

Los métodos a añadir a la clase `Vector` son:

- Método `grabaEnFichero`: graba el vector actual en un fichero binario de tipos primitivos (NO usar el método `writeObject` para ficheros binarios). El nombre del fichero se pasa al método como parámetro.
- Un constructor de la clase `Vector`: recibe como parámetro el nombre del fichero (creado con `grabaEnFichero`) en el que se encuentra guardado el vector y crea un objeto de la clase.

Ambos métodos no deberán tratar las excepciones debidas a la gestión de los ficheros, sino que las deberán lanzar para que las pueda tratar el método de nivel superior.

El formato del fichero se deja a la elección del alumno.

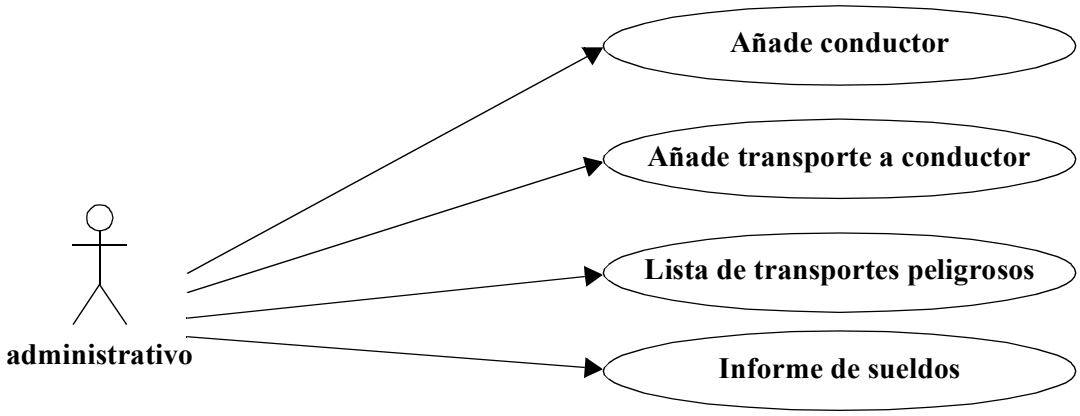
3) (4.25 puntos) Completa el diseño e implementación de la aplicación (únicamente de las partes solicitadas) que verifica los siguientes requisitos:

Se desea desarrollar una aplicación que permita a una empresa de transporte de mercancías y personas llevar la cuenta del sueldo a pagar a sus conductores. Cada conductor se identifica por su DNI. El sueldo de cada conductor depende del número de horas conducidas y del tipo de transportes realizados.

Cada conductor cobra un sueldo base fijo de 700€ al que hay que sumar 5€ por cada hora conducida en cualquier tipo de transporte. Además según el transporte existen unos extras:

- Transporte de personas: extra de un 1€ por hora si se trata de un transporte colectivo (el número de personas transportadas es mayor de 9) y 0.5€ si no es transporte colectivo.
- Transportes de mercancías: extra de 2€ por tonelada transportada.
- Transporte de mercancías peligrosas: igual que el transporte de mercancías más un fijo extra de 50€ por cada transporte realizado.

La aplicación deberá implementar los siguientes casos de uso:



(Una aplicación real incluiría también casos de uso para eliminar conductores, iniciar el sueldo acumulado al principio del mes, etc. No se incluyen en el problema para limitar su complejidad)

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

1. El administrativo elige la opción "Añade conductor" e introduce el DNI del nuevo conductor.
2. La aplicación añade el conductor a la plantilla de la empresa.
   - En el caso de que ya exista un conductor con ese DNI se notifica y no se añade.

Caso de uso "Añade transporte a conductor":

1a. El administrativo elige la opción "Añade transporte a conductor" e introduce el DNI del conductor
   - En el caso de que no haya ningún conductor con ese DNI se notifica y finaliza el caso de uso

1b. El administrativo introduce los datos correspondientes al tipo de transporte realizado

2. La aplicación añade el transporte al conductor

Caso de uso "Lista de transportes peligrosos":

1. El administrativo elige la opción "Lista de transportes peligrosos" e introduce el DNI del conductor
2. La aplicación muestra los transportes de mercancías peligrosas realizados por ese conductor

Caso de uso "Informe de sueldos":

1. El administrativo elige la opción "Informe de sueldos".
2. La aplicación muestra por consola los sueldos de todos los conductores. El formato debe ser exactamente el mostrado a continuación (respetando la ordenación por columnas, número de decimales, etc.).

```
DNI: 12345678A  Sueldo: 1234.56
DNI: 23456789B  Sueldo:  700.00
DNI: 34567890C  Sueldo: 2100.12
```

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 administrativo 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 listas de conductores, la asignación de transportes a conductores, etc.).

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 y el polimorfismo)