Lenguajes de Programación

ETSI Informática, UNED

Curso 2005/2006

Práctica

"Reconocedor de Nombres Propios"

Introducción

Los objetivos que se plantean en la realización de esta práctica son los siguientes:

- Familiarización con la programación orientada a objetos (POO): definición de clases e instancias, uso de la herencia, definición/uso de métodos estáticos y abstractos,
- Realización del diseño orientado a objetos de un problema,

Implementación de un programa sencillo donde se manejen conceptos relacionados con POO. La práctica se va a implementar en Java 2 Standard Edition (J2SE 1.5) [1]. El compilador de Java está incluido puede descargarse en http://java.sun.com/j2se/1.5/. La asignatura dispone de una dirección en Internet en la que se incluye un apartado dedicado a prácticas. Aquí se incluye información relativa a la planificación y seguimiento de las prácticas, así como el enunciado. La dirección Internet es la siguiente http://www.lsi.uned.es/lp/practica.html.

Programación Orientada a Objetos en Java

El paradigma de programación orientada a objetos define un programa como una colección de entidades que se relacionan para resolver un problema. Estas entidades, que se conocen genéricamente como objetos, están tienen por un conjunto de propiedades y métodos, y están organizadas en torno a una jerarquía de clases.

En Java cada objeto puede tener variables y métodos privados y públicos. Se puede modificar dicha visibilidad de una clase usando *los modificadores de acceso* a miembros. Las dos maneras más habituales de especificar la accesibilidad son:

private – la variable o método está disponible solamente para esta clase,

public – la variable o método está disponible para todas las clases,

Una clase puede heredar los variables y métodos públicos de otra clase a través del mecanismo de herencia y la palabra clave **extends**. Por ejemplo:

```
//una clase base que va a contener información sobre los vehículos de
nuestra empresa:
public vehiculo {
         private int noPuertas;
         private int noRuedas;
         private String modelo;
         public vehiculo(){}
         public void setNoPuertas(int np) {
                  noPuertas = np;
      }
      //etc.
}
//una clase para tratar a los coches en general...
public coche extends vehiculo {
         private
         private boolean airbags;
         public coche(){}
         public void setAirbags(Boolean a) {
                  airbags = a;
      }
      //etc.
}
//y, por fin, una clase para tratar a los coches deportivos
public final cocheDeportivo extends vehículo {
         private String capacidadMotor;
         private int maxVelocidad;
         public cocheDeportivo(){}
         public void setCapacidadMotor(String cm) {
                  capacidadMotor = cm;
         }
         //etc.
         //se puede llamar a cualquier método en las superclases como
si estuvieran dentro
      //de esta misma clase, p.ej.:
      setNoPuertas(2);
}
```

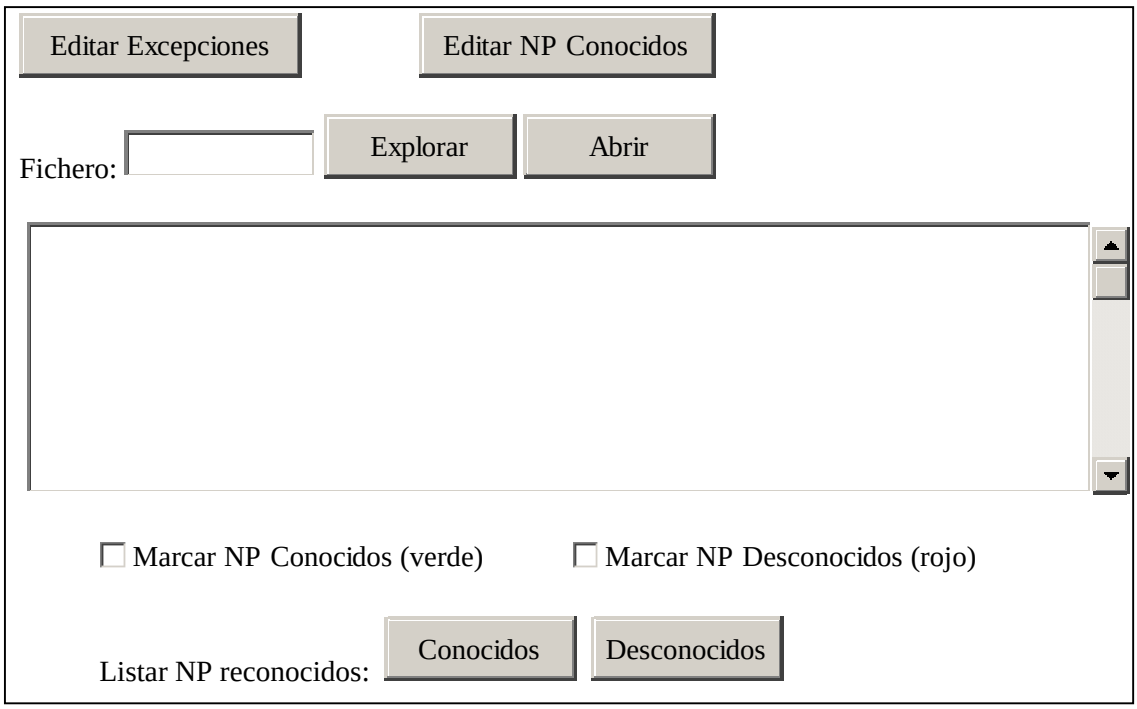
Notas: Las clases que extienden otras clases tienen el nombre de subclases y las clases que son extendidas por otras clases tienen el nombre de superclases.

Hay que tener cuidado a la hora de planificar las relaciones de herencia entre clases en Java porque una clase solamente puede heredar variables y métodos de otra (y sus superclases). Es decir, que no hay herencia múltiple en Java como hay en lenguajes como C++ (aunque se puede reproducir la técnica de herencia múltiple usando interfaces…). De todas formas, la manera más habitual para tratar está tema es simplemente usar una clase dentro de otra, por ejemplo, si hay una clase para el aparcamiento de una empresa que ya es una extensión de una clase base aparcamiento, dicha clase no puede heredar ninguna otra clase, por lo tanto, se incluirán las clases de coches, camiones, motos, etc., así:

```
public aparcamientoEmpresa extends aparcamiento {
         private String nombreEmpresa;
         private cocheDirector = new cocheDeportivo(…);
         public aparcamientoEmpresa(){}
         //etc.
         //para llamar a algún método en una clase hay que especificar
la variable de
      //la instancia...
      cocheDirector.setCapacidadMotor(“4.5l”);
}
```

Descripción de la práctica

Se trata de implementar un programa que realice la búsqueda de Nombres Propios en un fichero. Un Nombre Propio se define como una secuencia de palabras que comienzan por mayúscula, y que no están presentes en un fichero de Excepciones. Los Nombres Propios pueden ser Conocidos o Desconocidos dependiendo respectivamente de que estén o no en un fichero de Nombres Propios Conocidos. El sistema deberá tener un interfaz gráfico cuya ventana principal sea similar a la de la siguiente figura:



Las funcionalidades de los distintos elementos del interfaz se detallan a continuación:

- *Editar Excepciones*: Se abre una ventana con la lista de Excepciones que contiene el correspondiente fichero de excepciones. La ventana debe permite añadir o eliminar Excepciones de forma que al volver a la ventana principal se actualice correctamente el fichero de excepciones.
- *Editar NP Conocidos*: Análogo al anterior pero respecto al fichero de NP conocidos.
- *Explorar Fichero*: Se abre el diálogo para indicar el fichero que se va a leer y en el que se reconocerán los Nombres Propios.
- *Abrir*: Abre el fichero seleccionado y muestra el texto en el área de texto.
- *Marcar NP Conocidos*: Marca en color verde los Nombres Propios conocidos encontrados en el texto.
- *Marcar NP Desconocidos*: Marca en color rojo los Nombres Propios del texto que no están en la lista de conocidos.
- *Listar NP reconocidos, Conocidos*: Se abre una nueva ventana donde se listan únicamente los Nombres Propios conocidos que han sido reconocidos en el texto.
- *Listar NP reconocidos, Desconocidos*: Se abre una nueva ventana donde se listan los Nombres Propios desconocidos que han sido reconocidos en el texto. En este caso, se pueden marcar uno o varios Nombres Propios de la lista y pulsar un botón de “Añadir a NP Conocidos”. El efecto es que se actualiza la lista (y el fichero) de Nombres Propios

Conocidos, añadiendo los que han sido marcados. Además, deberá modificarse adecuadamente la visualización de NP reconocidos que son conocidos (texto y lista).

Para aprender el uso de las clases de java correspondientes a los elementos del interfaz, se recomienda consultar los ejemplos disponibles con el SDK (del J2SE).

Plan de trabajo

Para realizar la práctica se seguirá el siguiente método de trabajo:

- En primer lugar se leerá detenidamente el enunciado de esta práctica
- A continuación hay que diseñar, utilizando un paradigma orientado a objetos, los elementos necesarios para la aplicación explicada en el apartado anterior. Debe hacerse uso de los mecanismos de herencia siempre que sea posible. Se valorará un buen diseño que favorezca la reutilización de código y facilite su mantenimiento.
- El código estará debidamente comentado.
- La clase principal que abre la aplicación deberá llamarse "ner.class"

Material a Entregar

- *Memoria*: La memoria constará de los siguientes apartados:
  - Portada con título "Práctica de Lenguajes de Programación – Junio 2006" y los datos del alumno: Nombre, Apellidos, dirección de correo electrónico y teléfono.
  - Diagrama de clases, detallando claramente el tipo de relación entre ellas (uso, agregación, herencia, ...).
  - Un texto en el que se describa cada objeto, justificación de su existencia, métodos públicos que contiene y funcionalidad que realizan.
  - Anexo con el código fuente de las clases implementadas.

- *CD*: incluyendo todos los ficheros *.java y *.class, así como la memoria en formato electrónico (preferiblemente html o pdf). El disquete estará libre de virus. **No se corregirá ni se tendrá en cuenta ninguna práctica que esté infectada por un virus**.

Normas de realización de la práctica

1. La realización de la práctica es obligatoria. Sólo se evaluará el examen si la práctica ha sido previamente aprobada.
2. Después de un trabajo inicial de diseño, las prácticas se hacen en los centros asociados supervisadas por el tutor de la asignatura. El trabajo a realizar con la práctica tiene dos partes: el diseño y la implementación. Cada alumno deberá llevar su diseño el día en que se realicen las prácticas para que el tutor lo supervise y el alumno puede realizar la implementación.
3. La práctica es individual. Las prácticas cuyo código coincida total o parcialmente con el de otro alumno serán motivo de suspenso para todos los implicados (copiadores y copiados), no pudiéndose examinar ninguno de ellos en el presente curso académico.
4. Cada tutor establecerá unas fechas para la realización de la práctica. El tutor remitirá, **antes del 2 de junio**, a la asignatura de Lenguajes de Programación un informe de calificación de cada alumno.

5. No habrá sesión extraordinaria de prácticas ya que la asignatura ya debe estar implantada en todos los centros asociados. En caso de que algún alumno no tuviera tutor, deberá dirigirse a cualquier otro centro asociado donde se imparta la asignatura.
6. El equipo docente tendrá en cuenta prácticas con notas altas para aquellos alumnos cuyo examen esté cercano al aprobado.
7. El alumno debería dirigirse a su tutor para cualquier duda que tenga sobre su práctica y solamente al equipo docente (por correo electrónico) en el caso de que su tutor no pueda resolver su problema. En este caso pedimos al alumno que, además de sus datos personales, nos envíe el nombre del centro asociado en el que está matriculado y el de su tutor.
8. Evidentemente se puede usar los foros para realizar consultas a los compañeros pero nunca para intercambiar código.

Normas para los Tutores

Como se puede apreciar, el papel del tutor es fundamental en todos los aspectos de la práctica tanto el planteamiento del problema, el diseño OO del programa, su desarrollo y su depuración. Tratándose de una asignatura obligatoria, cada alumno debería tener acceso a un tutor.

Los tutores deben seguir los siguientes pasos:

1. Enviar por fax sus datos personales, nombre del centro, nombre de usuario y clave de acceso que desee utilizar para la aplicación de calificaciones, a la atención del Equipo Docente de Lenguajes de Programación, FAX: 91 398 65 35. La hoja deberá estar firmada por el tutor y sellada por el centro.
2. Indicar a los alumnos que deben darse de alta en la aplicación de calificaciones a través del enlace correspondiente que encontrarán dentro del entorno virtual. El alta consistirá en seleccionar centro y tutor e introducir su DNI.
3. Una vez terminada y entregada la práctica, el tutor tiene que evaluar el diseño OO del programa y su funcionamiento, calificar la práctica e introducir las calificaciones en la aplicación disponible en http://www.lsi.uned.es/lp/tutores.
4. Emitir los listados de calificación por medio de la aplicación y enviarlos firmados y sellados por el centro a la atención de:

   Equipo Docente de Lenguajes de Programación
   Dpto. Lenguajes y Sistemas Informáticos
   ETS Ingeniería Informática, UNED
   Juan del Rosal, 16
   28040 Madrid

5. Comunicar la calificación a sus alumnos.

El papel de los Centros Asociados en las prácticas de asignaturas obligatorias

Las prácticas son esenciales en las titulaciones de Informática porque, entre otras cosas, permiten a los alumnos adquirir conocimientos importantes sobre los aspectos más aplicados de ciertas asignaturas, lo cual resulta de gran relevancia e interés a la hora de acceder a un puesto laboral relacionado con la Informática. Por ello, las prácticas son obligatorias en las asignaturas que imparten los equipos docentes del Departamento de Lenguajes y Sistemas Informáticos (LSI).

Para orientar y ayudar a los alumnos, así como para comprobar que realmente un alumno ha realizado su práctica de forma satisfactoria, ésta se debe realizar en un Centro Asociado bajo la supervisión de un tutor.

Un Centro Asociado que ha permitido a un alumno matricularse en una asignatura obligatoria de una carrera de Informática debería ayudarle a encontrar una solución al problema de la realización de las prácticas. Si se trata de una asignatura donde no se han matriculado muchos alumnos, quizás el centro no cuente con recursos para proporcionar un tutor específicamente para la asignatura. Si hay otro Centro Asociado muy cerca que dispone de tutor, quizás el alumno pueda realizar la práctica allí pero si no es así, el Centro Asociado debería proporcionar un tutor para supervisar y corregir las prácticas de sus alumnos. Lo más razonable sería que fuera un tutor de otra asignatura de Informática en el mismo Centro el que hiciera la sesión de prácticas para los alumnos de la asignatura en cuestión, y al final de la sesión evaluara los trabajos de los alumnos, según las pautas marcadas por el equipo docente, haciendo llegar a éste las calificaciones otorgadas.

De vez en cuando sucede que un alumno se pone en contacto con un equipo docente del Departamento de LSI porque se ha matriculado en una asignatura obligatoria en un Centro Asociado que no le proporciona un tutor para supervisar la práctica, ¡aunque le ha permitido matricularse! Lo que quiere el alumno es que el equipo docente le proporcione una solución a este problema, como por ejemplo, la posibilidad de asistir a unas sesiones extraordinarias de prácticas en la Sede Central de la UNED en Madrid o la posibilidad de realizar la práctica por su cuenta en casa, enviándola a continuación al equipo docente para su corrección. Sin embargo, los equipos docentes de LSI no disponen de recursos para ninguna de estas dos alternativas.

Por lo tanto, un alumno que tras haberse matriculado en una asignatura obligatoria en un Centro Asociado, se encuentre con que el centro no tiene tutor para dicha asignatura, debería dirigirse al director del Centro para solicitar de él una solución, tal como se ha presentado aquí, es decir, alguien que pueda supervisar y corregir su práctica con plenas garantías.

En el caso de que el director no le proporcione una solución, el alumno debería comunicárselo, por escrito, lo antes posible, a la directora del Departamento de Lenguajes y Sistemas Informáticos, Dra. Mª Felisa Verdejo.

Referencias

[1] Java 2 API Specification http://java.sun.com/j2se/1.5/

[2] *The Java Tutorial*, http://java.sun.com/docs/books/tutorial/