Prácticas POO
Curso 08/09

Alejandro Bellogín

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Marzo 2009

http://www.eps.uam.es/~abellogin

Esquema

- IDE
- Cosas de Java (útiles para esta práctica)
- Organización P2:
  - Semana 1
  - Semana 2
  - Semana 3
  - Apariencia estimada al final de esta práctica
- Más cosas de Java

IDE

- Deberíais saber
  - Crear proyectos (desde cero y con código)
  - Ejecutar
  - Depurar (breakpoint, watch, variables locales)
  - Lanzar tareas de un build.xml (targets)

  - Entender build.xml y hacer pequeñas modificaciones

Más Java

- Útil para esta práctica:
  - Control de acceso
  - ArrayList
  - HashMap
  - Enhanced for
  - Hilos
  - Excepciones
  - Entrada / salida
  - ObjectInputStream / ObjectOutputStream

Control de acceso

- Ocultación de
  - Variables
  - Métodos
  - Constructores

- Todas variables public ==> mal implementado (normalmente)

ArrayList

- Conjunto variable de cualquier tipo de objetos
- Similar a array, pero su capacidad aumenta o disminuye **dinámicamente**

- Desde 1.5: arrays tipados

```
protected ArrayList<Alumno> alumnos;
```

(en tiempo de compilación nos aseguramos el tipo del contenido)

HashMap

- Manera sencilla de tener una tabla (hash)

- Desde 1.5: tablas tipadas

```
private HashMap<String, Alumno> alumnos;
// ...
alumnos.put(a.getDni(), a);
```

Enhanced for

- Tipos de iteración:
  - Con iterador (clase Iterator)
  - Sin iterador (usando una variable como índice)
  - Enhanced for

```
for (Alumno alumno : alumnos) {
    for (AsignaturaTeoria at : alumno.getExpediente().getTodasAsignaturasComoArray()) {

    }
}
```

Hilos

- El intérprete de Java hace un uso intensivo de hilos.
- Esto provoca situaciones raras:

```
Escribe su dirección:
Escribe su teléfono:
aaa
aaaa
```

- Veremos más cosas en la P3 (GUI)

Excepciones

- Cómo (y cuándo) lanzarlas

```
public void cerrarExpediente() throws IllegalArgumentException {
    if (!esPosibleCerrarExpediente()) {
        throw new IllegalArgumentException("No es posible cerrar el expediente");
    }
```

  - Throws: en la definición del método
  - Throw: dentro del método

- Cómo (y cuándo) tratarlas

```
try {
    cerrarExpediente();
} catch (IllegalArgumentException e) {
    // nada
}
```

  - Try-catch: se capturan las que se quieran, las demás se lanzan

Entrada / salida

- En esta práctica:

```
public static String leeLineaDeTeclado() {
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
    String s = null;
    try {
        s = entrada.readLine();
    } catch (Exception e) {
    }
    return s;
}
```

```
private static void imprimirMenu() {
    System.out.println("Elige una de estas opciones:");
    System.out.println("0: Salir del sistema");
    System.out.println("1: Crear un alumno");
    System.out.println("2: Matricular a un alumno determinado de ur
    System.out.println("3: Añadir una nota a una asignatura");
    System.out.println("4: Cerrar actas");
    System.out.println("5: Cerrar expediente");
    System.out.println("6: Generar informes de asignaturas (alumnos
    System.out.println("7: Generar informes de asignaturas (resumer
    System.out.println("8: Generar informes de alumnos (expediente
    System.out.println("9: Generar informes de alumnos (expediente
    System.out.println("10: Salvar el estado actual");
    System.out.println("11: Leer un sistema salvado previamente");
}
```

- En la siguiente ...

Guardar y cargar (un sistema)

```
public void guardar(String ficheroASalvar) throws FileNotFoundException,
        IOException {
    FileOutputStream fos = new FileOutputStream(ficheroASalvar);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(this);
    oos.close();
}

public static Sistema cargar(String ficheroACargar) throws FileNotFoundException,
        IOException, ClassNotFoundException {
    FileInputStream fis = new FileInputStream(ficheroACargar);
    ObjectInputStream ois = new ObjectInputStream(fis);
    Sistema s = (Sistema) ois.readObject();
    ois.close();
    return s;
}
```

Proyecto

- Aplicación que:
  - Crea alumnos
  - Matricula un alumno de n asignaturas
  - Cierra expedientes y calcula medias
  - Calcula calificación final de las asignaturas (usando teoría y práctica, si tiene)
  - Cierra actas (para la convocatoria y curso actual, permite ‘No consume’)
  - Genera informe de asignaturas y de alumnos

Es independiente de

- La forma en que se imprimen las cosas (menú)

- La manera en que se pide la entrada (nuevos alumnos, matriculaciones, notas, ...)

Práctica 2

- Implementar el diseño que se os ha entregado
- Cuidar la encapsulación de la entrada / salida
  - En la próxima práctica sólo deberíais reescribir esta parte

Semana 1

- Entender el diseño
- Ver qué parte del diseño (métodos, atributos) corresponde con cada funcionalidad
- Implementar funcionalidades, sin pensar en entrada/salida
  - El enfoque debe ser de aislar al máximo esta parte
  - Así la P3 será más fácil

Semana 2

- Terminar de implementar
- Hacer entrada / salida
- Comprobar si el resultado permite cambiar los métodos de entrada / salida fácilmente
- Hacer pruebas, depurar

Semana 3

- Una vez funciona la práctica, mirar la última transparencia (Aún más Java)

- Explicación de la P3 (a petición)

Captura final del sistema?

```
Elige una de estas opciones:
0: Salir del sistema
1: Crear un alumno
2: Matricular a un alumno determinado de una asignatura
3: Añadir una nota a una asignatura
4: Cerrar actas
5: Cerrar expediente
6: Generar informes de asignaturas (alumnos matriculados en el curso académico actual)
7: Generar informes de asignaturas (resumen de calificaciones)
8: Generar informes de alumnos (expediente completo del alumno)
9: Generar informes de alumnos (expediente del alumno)
10: Salvar el estado actual
11: Leer un sistema salvado previamente
```

(Aún) Más Java

- Patrones de diseño:
  - Singleton
  - Factory
  - ...
- Entrada / Salida alternativa: java.io.Console
- Funciones matemáticas: java.lang.Math