Prácticas POO
Curso 10/11

Alejandro Bellogín

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Febrero 2011

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

Esquema

- Parte I
  - Contacto
  - Organización de las prácticas
    - Normas
    - Calendario
- Parte II
  - Explicación de la práctica
  - Nociones básicas (SO, IDE, …)
  - Algunas cosas de Java
  - Más cosas de Java
- Parte III
  - A practicar

Esquema

- Parte I
  - Contacto
  - Organización de las prácticas
    - Normas
    - Calendario
- Parte II
  - Explicación de la práctica
  - Algo de Java
  - Más Java
- Parte III
  - A practicar

Contacto

- **Correo electrónico** (preferible: asunto '[poo]'):

  alejandro . bellogin @ uam . es

- Despacho: B – 408
- Hora de tutorías? ← acordadas previamente!
  - Tentativo: Miércoles de 18:00 a 20:00
- En clase:
  - lunes de 14:00 a 16:00
  - martes de 14:00 a 16:00
- Teléfono: 91 497 22 93

Normativa

- Para promediar es necesario aprobar independientemente la teoría y las prácticas. Se convalidarán con una nota de 5 las prácticas aprobadas de un año para el siguiente, siempre que se contase con un 3 o más en el examen de teoría correspondiente.
- Los grupos de trabajo serán de 2 personas
- La copia en prácticas es una falta grave que será objeto de sanción (que puede ser extensible al copiado): apertura de **expediente de expulsión** o bien **suspenso automático sin posibilidad de presentarse a Septiembre**.
- El intercambio de *ideas* no se considera copia (es más, se recomienda encarecidamente). El intercambio de código fuente sí, y se castigará como tal. El estudiante es responsable de evitar que su material evaluable (código, problemas, ejercicios, memorias de prácticas, etc.) sea accesible a estudiantes de otros grupos de prácticas.
- Para aprobar la asignatura, es obligatorio haber entregado todas las prácticas.
- Los alumnos que entreguen en **lunes** deben hacerlo **como tarde 2 horas antes** del comienzo de la clase de prácticas. Aquellos alumnos que entreguen **cualquier otro día** de la semana tienen como plazo **las 23:59 del día anterior**. Los retrasos dentro del mismo día de entrega descontarán un 20% de la nota. A partir del primer día de retraso, cada día sucesivo resta otro 10% del total, llegándose al 100% a los 8 días de cumplirse el plazo. No se considera como entrega aquella que sólo contiene código o sólo contiene la memoria.
- La última semana de prácticas se reserva para un examen de prácticas para aquellos alumnos que no han podido asistir a alguna revisión, con entregas que no funcionan, etcétera.
- Las prácticas se enviarán como un único fichero mediante el sistema de entrega de prácticas de la Escuela. Nombre: p<*número de práctica*><*letra de grupo*><*número de pareja, 2 dígitos*>.zip

http://arantxa.ii.uam.es/~poo/practicas/normas.html

Normativa

- Para promediar es necesario aprobar independientemente la teoría y las prácticas. Se convalidarán con una nota de 5 las prácticas aprobadas de un año para el siguiente, siempre que se contase con un 3 o más en el examen de teoría correspondiente.
- Los grupos de trabajo serán de 2 personas
- La copia en prácticas es una falta grave que será objeto de sanción (que puede ser extensible al copiado): apertura de **expediente de expulsión** o bien **suspenso automático sin posibilidad de presentarse a Septiembre**.
- El intercambio de ideas no se considera copia (es más, se recomienda encarecidamente). El intercambio de código fuente sí, y se castigará como tal. El estudiante es responsable de evitar que su material evaluable (código, problemas, ejercicios, memorias de prácticas, etc.) sea accesible a estudiantes de cursos posteriores.
- Para aprobar la asignatura, es obligatorio haber entregado todas las prácticas.
- Los alumnos que entreguen en Labs deben hacerlo como tarde **2 horas antes** del comienzo de la clase de prácticas. Aquellos alumnos que entreguen **cualquier otro día** de la semana tienen como plazo **las 23:59 del día anterior**. Los retrasos dentro del mismo día de entrega descontarán un 20% de la nota. A partir del primer día de retraso, cada día sucesivo resta otro 10% del total, llegándose al 100% a los 8 días de cumplirse el plazo. No se considera como entrega aquella que sólo contiene código o sólo contiene la memoria.
- La última semana de prácticas se reserva para un examen de prácticas para aquellos alumnos que no han podido asistir a alguna revisión, con entregas que no funcionan, etcétera.
- Las prácticas se enviarán como un único fichero mediante el sistema de entrega de prácticas de la Escuela. Nombre: p<*número de práctica*><*letra de grupo*><*número de pareja, 2 dígitos*>.zip

- No copiar
- Entregar antes de las 23:59, los lunes
- Asistencia obligatoria el día del examen

http://arantxa.ii.uam.es/~poo/practicas/normas.html

Prácticas

- Tres prácticas:
  - P1 (20%, 2 semanas) : introducción (diseño, primeras clases)
  - P2 (40%, 3 sem): clases, herencia
  - P3 (40%, 4-5 sem): interfaces gráficas, librerías

- Las prácticas se corrigen mediante inspección de lo entregado y con un *examen presencial* de cada práctica.
- 40% de la nota final

Más información:
http://arantxa.ii.uam.es/~poo/practicas/normas.html

Calendario

Lunes D	**Martes** **F, E**	Miercoles C	Jueves B	Viernes A	Práctica/Explicaciones
			17 Feb. Inicio P1	18 Feb. Inicio P1	Inicio P1: introducción a Java; incluye diseño e introducción al entorno
21 Feb. Inicio P1	**22 Feb.** **Inicio P1**	23 Feb. Inicio P1	24 Feb.	25 Feb.	
28 Feb.	**01 Mar.**	02 Mar.	03 Mar. Evaluación P1 Inicio P2	04 Mar. Evaluación P1 Inicio P2	Entrega P1 Inicio P2: Clases y Herencia
07 Mar. Evaluación P1 Inicio P2	**08 Mar.** **Evaluación P1** **Inicio P2**	09 Mar. Evaluación P1 Inicio P2	10 Mar.	11 Mar.	
14 Mar.	**15 Mar.**	16 Mar.	17 Mar.	18 Mar.	
21 Mar.	**22 Mar.**	23 Mar.	24 Mar. Parciales	25 Mar. Parciales	
28 Mar. Evaluación P2 Inicio P3	**29 Mar.** **Evaluación P2** **Inicio P3**	30 Mar. Evaluación P2 Inicio P3	31 Mar. Evaluación P2 Inicio P3	01 Abr. Evaluación P2 Inicio P3	Entrega P2, grupos A y B entregan el lunes (como el grupo D) Inicio P3: Interfaces Gráficas y uso de librerías
04 Abr.	**05 Abr.**	06 Abr.	07 Abr.	08 Abr.	
11 Abr.	**12 Abr.**	13 Abr.	14 Abr.	15 Abr.	
18 Abr. Semana Santa	**19 Abr.** **Semana Santa**	20 Abr. Semana Santa	21 Abr. Semana Santa	22 Abr. Semana Santa	Vacaciones de Semana Santa
25 Abr. Semana Santa	**26 Abr.**	27 Abr.	28 Abr.	29 Abr.	
02 May. Festivo	**03 May.**	04 May.	05 May.	06 May.	
09 Abr.	**10 May.** **Evaluación P3**	11 May. Evaluación P3	12 May. Evaluación P3	13 May. Fiesta EPS	Entrega P3, grupo A entrega el lunes (como el grupo D)
16 May. Evaluación P3	**17 May.** **Evaluación**	18 May. Evaluación	19 May. Evaluación	20 May. Evaluación	Evaluaciones extraordinarias

Esquema

- Parte I
  - Contacto
  - Organización de las prácticas
    - Normas
    - Calendario
- Parte II
  - Explicación de la práctica
  - Algo de Java
  - Más Java
- Parte III
  - A practicar

Práctica 1

- Analizar el problema que se plantea
  - *Tienda que vende artículos a clientes*
- Probar main entregado
- Implementar clases según diagrama UML
- Confirmar que funciona correctamente
  - Mínimo: que el main de prueba funcione y devuelva la salida esperada

Conceptos necesarios para la P1

- Entender diagrama de clases UML
- Noción de paquete (*package*) en Java
- Constructores
- Atributos, métodos (esp. *main(), toString()*)
- Javadoc

UML (diagrama de clases)

- Relaciones entre clases:
  - Asociación 
  - Agregación (“es parte de …”) 
  - Generalización o herencia (“es un caso particular de …”) 
- Navegabilidad: unidireccional vs bidireccional 
- Pueden tener nombres (roles)

UML en Java (I)

- Relación entre clases
  - Una es atributo de otra
  - La cardinalidad indica arrays
- Navegabilidad
  - Una conoce a la otra, viceversa, o ambas
- Roles
  - Normalmente, asociados a métodos

```java
public class Arbol {

    private Subarbol[] componentes;

    public Subarbol[] getComponentes() {
        return componentes;
    }
}
```

```java
public class Subarbol {

    private Arbol padre;

    public Subarbol(Arbol padre) {
        this.padre = padre;
    }

    public Arbol getPadre() {
        return padre;
    }
}
```

UML en Java (II)

- Interfaces
- Herencia
- Clases abstractas
  - Tipo especial de herencia, donde se definen métodos (que pueden ser llamados) pero no se implementan

```java
public interface Dibujo {

    public void resize();
}
```

```java
public abstract class Figura {

    public abstract double calculaArea();

    @Override
    public String toString() {
        return "Figura con área " + calculaArea();
    }
}
```

```java
public class Circulo extends Figura {

    private double radio;

    public double calculaArea() {
        return Math.PI * radio * radio;
    }
}
```

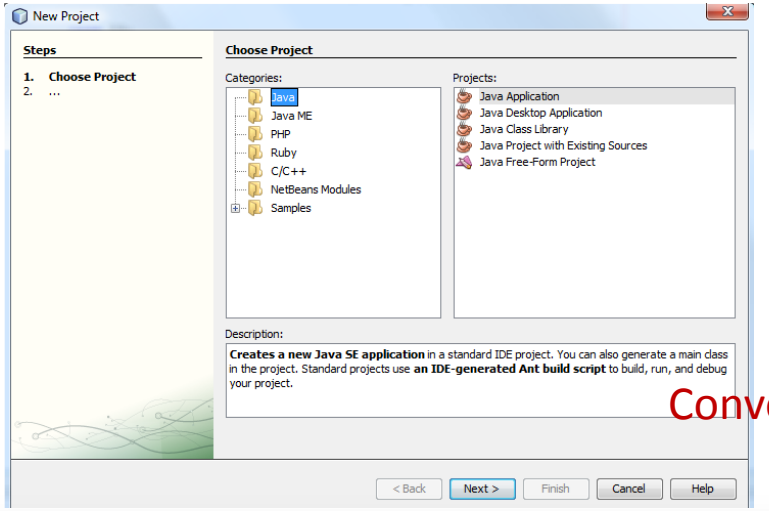
Conceptos Java que veremos en otras prácticas

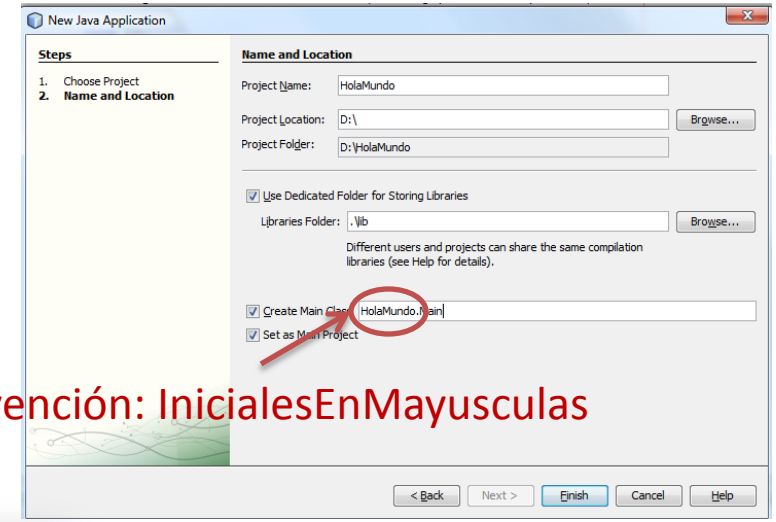
- Herencia
- Manejo de excepciones
- Hilos
- Interfaces gráficas
- Uso de librerías externas
- Programación distribuida

Esquema

- Parte I
  - Contacto
  - Organización de las prácticas
    - Normas
    - Calendario
- Parte II
  - Explicación de la práctica
  - Algo de Java
  - Más Java
- Parte III
  - A practicar

Practica: ‘Hola Mundo’ con NetBeans

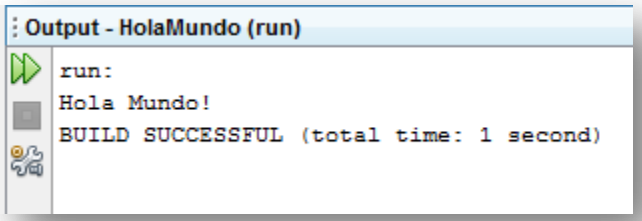




Convención: InicialesEnMayusculas

```
6   package HolaMundo;
7
8   /**
9    *
10   * @author Alejandro
11   */
12  public class Main {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          System.out.println("Hola Mundo!");
19      }
20
21  }
22
```

+ F6 =



Practica: depura con NetBeans

```java
package HolaMundo;

/**
 *
 * @author Alejandro
 */
public class Main {

    static int prueba = 1;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        System.out.println("Hola Mundo!");

        prueba = 2;

        System.out.println("Adiós Mundo!");
    }
}
```

añadir breakpoints

```java
package HolaMundo;

/**
 *
 * @author Alejandro
 */
public class Main {

    static int prueba = 1;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        System.out.println("Hola Mundo!");

        prueba = 2;

        System.out.println("Adiós Mundo!");
    }
}
```

+ Ctrl+F5

```java
13      static int prueba = 1;
14
15      /**
16       * @param args the command line arguments
17       */
18      public static void main(String[] args) {
19
20          System.out.println("Hola Mundo!");
21
22          prueba = 2;
23
24          System.out.println("Adiós Mundo!");
25      }
```

20:1 INS

Output

Name	Type	Value
Static		
class	Class	class HolaMundo.Main
prueba	int	1
args	String[]	#43(length=0)

Variables locales

+F5

```java
    static int prueba = 1;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        System.out.println("Hola Mundo!");

        prueba = 2;

        System.out.println("Adiós Mundo!");
    }
}
```

4:1 INS

tput

Name	Type	Value
Static		
class	Class	class HolaMundo.Main
prueba	int	2
args	String[]	#43(length=0)

El valor de esta variable ha cambiado!

+ F5
Salida

Output

HolaMundo (debug)

```
debug:
Hola Mundo!
Adiós Mundo!
```

FIN

Más cosas de NetBeans

- A veces parece que NetBeans programa solo:
  - Atajos de teclado (combinaciones de teclas, completar código)
  - Ingeniería Inversa
  - …

Alt+Shift+F	Formatear código
Ctrl+R	Renombrar (+ refactor!)
Alt+Shift+C	Comentar
‘sout’ + tab/espacio	‘System.out.println()’
‘re’ + tab/espacio	‘return ‘
…	…

- Integrado un Profiler (similar al Valgrind): analiza memoria y performance
- Viene con servidor de aplicaciones (Tomcat)
- CVS, SVN