Universidad Autónoma de Madrid
Escuela Politécnica Superior

Análisis y Diseño de Software (ADSOF)

Curso 2017-2018

Normativa de prácticas

Iván Cantador

ivan.cantador@uam.es

Aspectos generales

- Las prácticas se realizan en parejas
- A lo largo del curso se realizarán 5 prácticas
  - P1: Introducción a Java ~1 semana
  - P2: Introducción al Diseño Orientado a Objetos - UML ~2 semanas
  - P3: Introducción a la Programación Orientada a Objetos con Java ~3 semanas
  - P4: Herencia, interfaces y excepciones en Java ~3 semanas (+ Semana Santa)
  - P5: Genericidad, Colecciones y Expresiones Lambda en Java y Patrones de Diseño ~3 semanas
- Existen dos tipos de evaluación:
  - Evaluación Continua (EC)
    - Asistencia a al menos el 85% de las clases
  - Evaluación Final o No Continua (EF)
    - Realización de un examen consistente en una práctica de mayor complejidad que las realizadas durante el curso
- Tanto en EC como en EF hay que entregar las prácticas en las fechas establecidas

Calificaciones

- La calificación de prácticas representa el 40% de la calificación de la asignatura
- La calificación final de prácticas se calculará sobre 10 puntos como :

$$P = 0{,}05*P1 + 0{,}15*P2 + 0{,}20*P3 + 0{,}30*P4 + 0{,}30*P5$$

donde PX es la calificación de la práctica X

- Para aprobar es necesario cumplir alguna de las siguientes condiciones:
  - EC: obtener al menos un 3,5 en cada práctica
  - EF: aprobar el examen de prácticas
- En caso de que no se cumplan las condiciones anteriores, se suspenderá la convocatoria ordinaria, y la nota final de prácticas será mínimo(4, P)

Entrega de prácticas

- Las prácticas se entregarán vía Moodle
- Las fechas de entrega serán:
  - Antes de las 8:45h del día de comienzo de la siguiente práctica - grupos del lunes
  - El día anterior al comienzo de la siguiente práctica - grupos de los martes, jueves y viernes
- La penalización por retraso en entrega será:
  - 1 día de retraso: 2 puntos de penalización
  - 2 días de retraso: 3 puntos de penalización
  - > 2 días de retraso: suspender las prácticas de la asignatura en la convocatoria ordinaria

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Análisis y Diseño de Software (ADSOF)

Curso 2017-2018

Conceptos básicos sobre Java

Clase = atributos + métodos

- En C:

```c
typedef struct
{
    float re;
    float im;
} Complejo;

Complejo *complejo_crear(float x, float y)
{
    Complejo *c;
    c = malloc(sizeof(Complejo));
    c->re = x;
    c->im = y;
    return c;
}

float modulo(Complejo *c)
{
   return sqrt(c->re*c->re +
               c->im*c->im);
}
```

- En Java:

```java
public class Complejo
{
   private float re;
   private float im;

   public Complejo(float x, float y)
   {
      this.re = x;
      this.im = y;
   }

   public float modulo()
   {
      return Math.sqrt(this.x*this.x +
                       this.y*this.y);
   }

} // fin Complejo
```

Programación Orientada a Objetos

- Al programar en **C** pensamos en **acciones/funciones**:
  - Acciones que se realizan sobre objetos (argumentos) de entrada

accion(objeto)

```
Complejo *c = complejo_crear(2.0, -5.0);
float m = modulo(c);
```

- Al programar en **Java** pensamos en **objetos**:
  - Objetos que realizan acciones

objeto.accion()

```
Complejo c = new Complejo(2.0, -5.0);
float m = c.modulo();
```

concatenar strings - main - println - toString

```
public class Complejo {
   private float re, im;

   public Complejo(float x, float y) {
      this.re = x;
      this.im = y;
   }

   public float modulo() {
      return Math.sqrt(this.x*this.x + this.y*this.y);
   }

   public String toString() {
      return "(" + this.re + ", " + this.im + ")";  // operador + concatena strings
   }

   public static void main(String args[]) {  // argumentos del programa en un String[]
      Complejo c = new Complejo(2, -3);
      System.out.println(c);  // equivalente a System.out.println(c.toString());
   }

} // fin Complejo
```

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Análisis y Diseño de Software (ADSOF)

Curso 2017-2018

Práctica 1

Introducción a Java

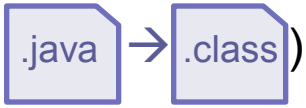
Inicio: semana del 5 de febrero

Duración: 1 semana

Entrega: semana del 12 de febrero

Peso en la calificación de prácticas: 5%

Práctica 1: Introducción a Java

- **Apartado 1: Hola Mundo**
  - Lenguaje Java
    - Uso de **package**
    - Implementación de una clase con un método **main**
    - Escritura en **stdout**
  - Programa javac.exe (compilador a *bytecode*: 

)
  - Programa java.exe (interprete)
    - Máquina Virtual de Java = **JVM** (*Java Virtual Machine*)
    - Existencia de su ruta en la variable de entorno PATH
    - Ejecución del interprete por línea de comandos
- **Apartado 2: Generación de documentación**
  - Programa javadoc.exe
  - Comentarios `/** */`
  - Etiquetas `@author`, `@version`, `@see`, `@param`, `@return`, `@exception`...

Práctica 1: Introducción a Java

- **Apartado 3: Números combinatorios** $\boxed{\binom{n}{k} = \frac{n!}{k!(n-k)!}}$
  - Clase **Combinatoria**
    - Cálculo de números combinatorios de forma recursiva
    - Constructor: Combinatoria(int n, int k)

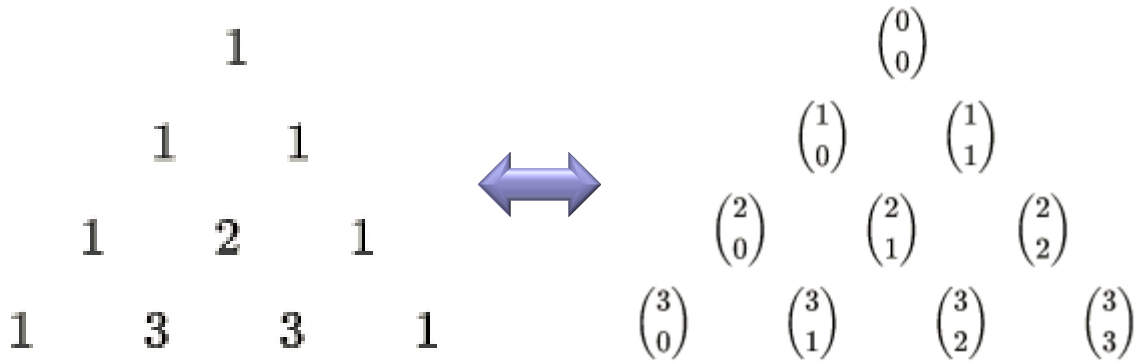
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad n,k>0 \qquad \binom{n}{0} = 1 \quad n \geq 0$$

$$\binom{0}{k} = 0 \quad k > 0$$

  - Lectura y procesamiento de argumentos de entrada de main
  - Llamada a métodos de una clase
  - Sintaxis básica Java
  - Ocurrencia de excepciones
    - El lanzamiento y la captura de excepciones se verá en prácticas posteriores

Práctica 1: Introducción a Java

- **Apartado 4**
  - Clase **Combinatoria**
    - Cálculo de números combinatorios $\binom{n}{k}$
    - Constructor: Combinatoria(int n, int k)
  - Clase **Tartaglia**
    - Visualización por pantalla del triángulo de Tartaglia / Pascal
    - Constructor: Tartaglia(Combinatoria c, int n)



Práctica 1: Introducción a Java

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

- **Apartado 5 (optativo)**
  - Modificación de la clase **Combinatoria**
    - Calcular y almacenar números combinatorios
    - En las expresiones recursivas evitar hacer cálculos ya realizados (almacenados)

posicionTartaglia(n, k) = n * (n + 1) / 2 + k

- Almacenar las posiciones en una caché de tipo tabla hash (de clave-valor)

```
Map<Integer, Long> cache = new HashMap<>();
...
int pos = posicion(n, k);
long valor;
...
cache.put(pos, valor);
if ( cache.containsKey(pos) )
   valor = cache.get(pos);
```

Array

Value
New York
Boston
Mexico
Kansas
Detroit
California

Hash Table

Key	Value
1	New York
2	Boston
3	Mexico
4	Kansas
5	Detroit
6	California