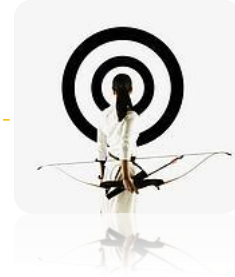
PSP0.1

Lección 4: Aprendiendo a medir software

Objetivos



- Entender los procesos PSP0.1 para:
  - Medir el tamaño de los programas que se producen
  - Realizar el conteo de tamaño para los programas que se producen
  - Realizar estimaciones de tamaño precisas y exactas
- Conocer cómo usar los scripts y formularios PSP0.1 que guían el proceso para:
  - Estar preparados para usar PSP0.1 para el siguiente programa

Universidad Carlos III De Madrid
Principios de Ingeniería Informática
María Isabel Sánchez Segura
José Arturo Mora-Soto
Juan Carlos Alonso Durán

Guión de PSP0.1

PSP0.1 Guión del Proceso

Propósito	Para guiar el desarrollo de programas a nivel de módulo
Criterios de Entrada	– Descripción del Problema – Formulario del Resumen del Plan del Proyecto PSP0.1 – Log de Registro de Tiempos y Defectos – Estándares de Tipo de Defecto, Codificación, y conteo del tamaño – Cronómetro (opcional)

Paso	Actividades	Descripción
1	Planificación	– Producir u obtener una declaración de requisitos. – ***Estimar el tamaño añadido y modificado.*** – Estimar el tiempo requerido de desarrollo. – Ingresar la información del Plan en el formulario Resumen del Plan de Proyecto. – Completar el Log de Registro de Tiempo.
2	Desarrollo	– Diseño del programa. – Implementar el diseño. – Compilar el programa, y reparar registrar todos los defectos encontrados. – Probar el programa, y arreglar y registrar todos los defectos encontrados. – Completar el Log de Registro del Tiempo.
3	Postmortem	– Completar el formulario del Resumen del Plan de Proyecto con la información de tamaños, defectos y tiempos actuales.

Criterios de Salida	– Un programa probado a profundidad – Formulario Resumen del Plan de Proyecto con los datos actuales y estimados. – ***Formularios PIP completados*** – Log de Registro de Tiempos y Defectos completado

Guión de PSP0.1

PSP0.1 Guión del Proceso

Propósito	Para guiar el desarrollo de programas a nivel de módulo
Criterios de Entrada	– Descripción del Problema – Formulario del Resumen del Plan del Proyecto PSP0.1 – Log de Registro de Tiempos y Defectos – Estándares de Tipo de Defecto, Codificación, y conteo del tamaño – Cronómetro (opcional)

Paso	Actividades	Descripción
1	Planificación	– Producir u obtener una declaración de requisitos. – ***Estimar el tamaño añadido y modificado.*** – Estimar el tiempo requerido de desarrollo. – Ingresar la información del Plan en el formulario Resumen del Plan de Proyecto. – Completar el Log de Registro de Tiempo.
2	Desarrollo	– Diseño del programa. – Implementar el diseño. – Compilar el programa, y reparar registrar todos los defectos encontrados. – Probar el programa, y arreglar y registrar todos los defectos encontrados. – Completar el Log de Registro del Tiempo.
3	Postmortem	– Completar el formulario del Resumen del Plan de Proyecto con la información de tamaños, defectos y tiempos actuales.

Criterios de Salida	– Un programa probado a profundidad – Formulario Resumen del Plan de Proyecto con los datos actuales y estimados. – ***Formularios PIP completados*** – Log de Registro de Tiempos y Defectos completado

PSP 0.1 Script – Planning Phase

PSP0.1 Guión de la Planificación

Propósito	Para guiar el proceso de Planificación del PSP
Criterios de Entrada	– Descripción del problema – Formulario resumen del Plan del Proyecto PSP0.1 – Log de Registro del Tiempo

Paso	Actividades	Descripción
1	Requisitos del Programa	– Producir u obtener una declaración de requisitos para el programa – Asegurar que la declaración de requisitos es claro y no ambiguo
2	Estimar Tamaño	– ***Realizar la mejor estimación del tamaño añadido y modificado del programa*** – ***Ingresar información del tamaño planificado en el formulario resumen del Plan de Proyecto.***
3	Estimar Recurso	– Realizar la mejor estimación del tiempo requerido para el desarrollo del programa – Ingresar información del Tiempo planificado en el formulario resumen del Plan del Proyecto – ***Usando el % a la fecha del programa desarrollado más recientemente, distribuir el tiempo de desarrollo sobre las fases del proyecto planificado.***

Criterios de Salida	– Declaración de requisitos documentados – Formulario Resumen del Plan del Proyecto completado con información del tamaño del programa estimado y tiempo de desarrollo – Log de Registro del Tiempo completado

PSP 0.1 Script – Planning Phase

PSP0.1 Guión de la Planificación

Propósito	Para guiar el proceso de Planificación del PSP
Criterios de Entrada	– Descripción del problema – Formulario resumen del Plan del Proyecto PSP0.1 – Log de Registro del Tiempo

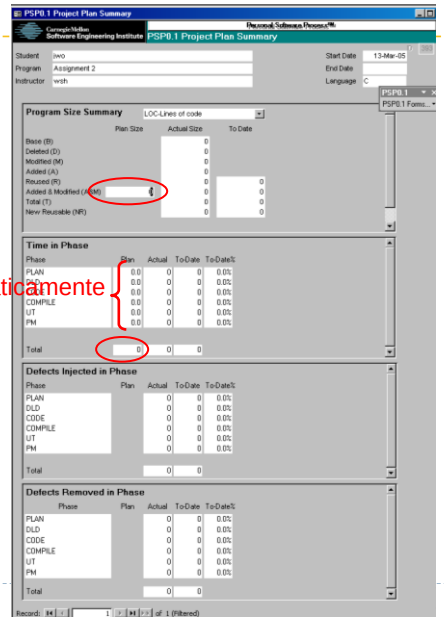
Paso	Actividades	Descripción
1	Requisitos del Programa	– Producir u obtener una declaración de requisitos para el programa – Asegurar que la declaración de requisitos es claro y no ambiguo
2	Estimar Tamaño	– ***Realizar la mejor estimación del tamaño añadido y modificado del programa*** – ***Ingresar información del tamaño planificado en el formulario resumen del Plan de Proyecto.***
3	Estimar Recurso	– Realizar la mejor estimación del tiempo requerido para el desarrollo del programa – Ingresar información del Tiempo planificado en el formulario resumen del Plan del Proyecto – ***Usando el % a la fecha del programa desarrollado más recientemente, distribuir el tiempo de desarrollo sobre las fases del proyecto planificado.***

Criterios de Salida	– Declaración de requisitos documentados – Formulario Resumen del Plan del Proyecto completado con información del tamaño del programa estimado y tiempo de desarrollo – Log de Registro del Tiempo completado

Esto lo hace Automáticamente La herramienta que Estamos usando

PSP0.1 Project Plan Summary

Se calcula automáticamente



PSP 0.1 Script – Development Phase

PSP0.1 Guión de Desarrollo

Propósito	Guiar el desarrollo de pequeños programas
Criterios de Entrada	– Declaración de requisitos – Formulario de resumen del Plan del Proyecto con el tamaño del programa y tiempo de desarrollo estimado – Log de Registro de Tiempos y Defectos – Estándar de Tipo de Defectos y *Estándar de Codificación*

Paso	Actividades	Descripción
1	Diseño	– Revisar los requisitos y producir un diseño para cumplirlos. – Registrar en el log de Registro de Tiempos cualquier defecto de requisito encontrado. – Registrar el tiempo en el Registro de Tiempos
2	Codificación	– Implementar el diseño ***siguiendo el Estándar de codificación*** – Registrar en el log de Registro de Defectos cualquier defecto de requisito o de diseño encontrado – Registrar el tiempo en el Log de Registro de Tiempos.
3	Compilación	– Compilar el programa hasta que esté libre de errores de compilación – Reparar todos los defectos encontrados – Registrar defectos en el Log de Registro de Defectos – Registrar el tiempo en el Log de Registro de Tiempos.
4	Prueba	– Probar hasta que todas las pruebas se ejecuten sin errores – Reparar todos los defectos encontrados – Registrar defectos en el Log de Registro de Defectos – Registrar el tiempo en el Log de Registro de Tiempos

Criterios de Salida	– Un programa probado completamente de acuerdo al ***Estándar de Codificación*** – Log de Registros de Tiempos y Defectos completados

PSP 0.1 Script – Development Phase

PSP0.1 Guión de Desarrollo

Propósito	Guiar el desarrollo de pequeños programas
Criterios de Entrada	– Declaración de requisitos – Formulario de resumen del Plan del Proyecto con el tamaño del programa y tiempo de desarrollo estimado – Log de Registro de Tiempos y Defectos – Estándar de Tipo de Defectos y ***Estándar de Codificación***

Paso	Actividades	Descripción
1	Diseño	– Revisar los requisitos y producir un diseño para cumplirlos. – Registrar en el log de Registro de Tiempos cualquier defecto de requisito encontrado. – Registrar el tiempo en el Registro de Tiempos
2	Codificación	– Implementar el diseño ***siguiendo el Estándar de codificación*** – Registrar en el log de Registro de Defectos cualquier defecto de requisito o de diseño encontrado – Registrar el tiempo en el Log de Registro de Tiempos.
3	Compilación	– Compilar el programa hasta que esté libre de errores de compilación – Reparar todos los defectos encontrados – Registrar defectos en el Log de Registro de Defectos – Registrar el tiempo en el Log de Registro de Tiempos.
4	Prueba	– Probar hasta que todas las pruebas se ejecuten sin errores – Reparar todos los defectos encontrados – Registrar defectos en el Log de Registro de Defectos – Registrar el tiempo en el Log de Registro de Tiempos

Criterios de Salida	– Un programa probado completamente de acuerdo al ***Estándar de Codificación*** – Log de Registros de Tiempos y Defectos completados

Genera un diseño que satisfaga los requisitos

- Es preciso que identifiqueis las clases, asigneis responsabilidades y decidais cómo implementar o modificarlas.
- Para eso recordar que podéis utilizar
  - Diagramas de secuencia de UML
  - Tarjetas CRC
- Un buen diseño:
  - Hará que el software que desarrolleis pase la fase de pruebas mas facilmente
  - Que tu programa no tenga lógica duplicada
  - Indica de forma precisa y clara cómo codificar el programa
  - Tiene el menor número de clases posible, pero recordar que tener una sola clase no es apropiado.

PSP 0.1 Script – Post-Mortem Phase

PSP0.1 Guión del Postmortem

Propósito	Para guiar el proceso postmortem del PSP
Criterios de Entrada	– Descripción del problema y declaración de requerimientos – Formulario Resumen del Plan del Proyecto con información del tamaño del programa y tiempo de desarrollo – Log de Registro de Defectos y Tiempos completados – Un programa probado y ejecutado de ***acuerdo a los Estándares de Codificación y conteo de tamaño***

Paso	Actividades	Descripción
1	Registrar defectos	– Revisar el Resumen del Plan del Proyecto para verificar que todos los defectos encontrados en cada fase fueron registrados. – Registrar algunos defectos omitidos
2	Consistencia de información de defectos	– Revisar que la información de cada defecto en el log de Registro de Defectos son precisos y completos – Verificar que los números de defectos introducidos y removidos por fase son razonables y correctos. – Corregir cualquier información de defecto omitido o incorrecto.
3	*Tamaño*	– ***Contabilizar el tamaño del programa completado*** – ***Determinar el tamaño de la código base, eliminado, modificado, reusado, total, y nuevo reusable*** – ***Ingresar esta información en el formulario Resumen del Plan del Proyecto***
4	Tiempo	– Revisar el log de Registro de Tiempos completado para ver si existen errores u omisiones – Corregir cualquier información de tiempo omitido o incompleto.

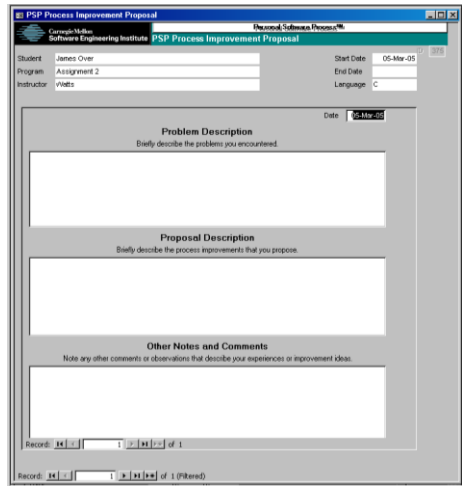
Criterios de Salida	– Un programa probado completamente de ***acuerdo a los Estándares de Codificación y conteo del tamaño*** – Formulario Resumen del Plan del Proyecto – ***Formulario PIP completado describiendo problemas del proceso, sugerencias de mejora y lecciones aprendidas*** – Log de Registros de Tiempos y Defectos completados

El formulario de propuesta de mejora del proceso (*process improvement proposal* – PIP)

- Para mejorar tu proceso, necesitas capturar problemas y proponer mejoras para referencias futuras.
- Necesitarás conocer
  - Cualquier problema que hayas encontrado al utiliza los procesos
  - Cualquier sugerencia que tengas para mejorar los procesos
  - Tus observaciones y resultados de realizar las prácticas

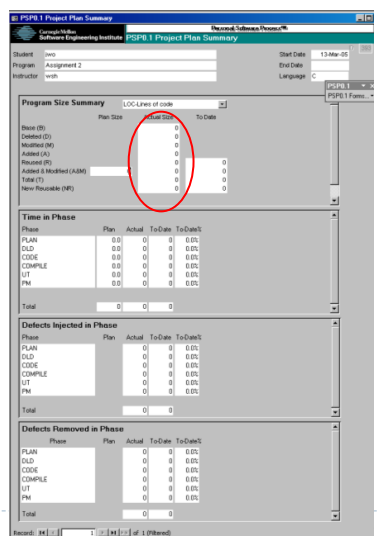
El formulario de propuesta de mejora del proceso (*process improvement proposal* – PIP)

- Debes completar un PIP para cada práctica
- PIP mantiene información sobre la mejora del proceso
  - fecha
  - Descripción del problema
  - Solución propuesta
  - Notas y comentarios

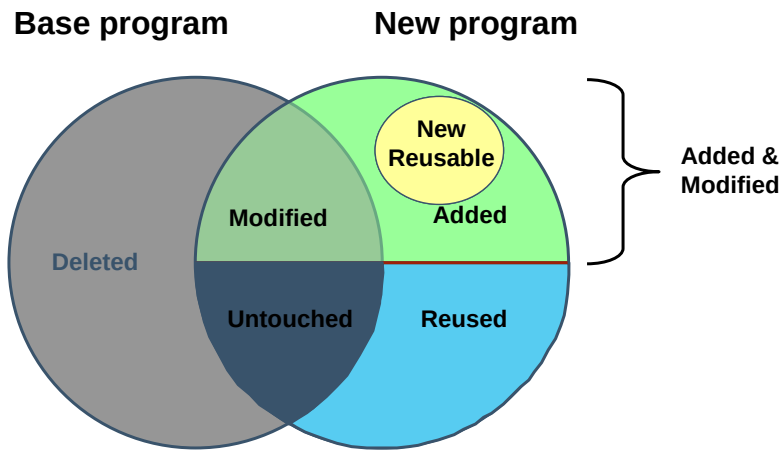


Resumen del plan de proyecto PSP0.1

- PSP0.1 añade el Resumen de Tamaño de Programa para los datos de tamaño estimado y real.
- Estos tipos de tamaño incluyen
  - base [B]
  - borrado [D]
  - modificado [M]
  - añadido [A]
  - reusado [R]
  - Añadido y modificado [A+M]
  - Nuevo reusable



Program Size Type Relationships



Estándar para cálculo de líneas de código

- Los elementos básicos del conteo de líneas de código son los siguientes:
  - Base: medida del tamaño del programa que tu estás tomando como base par el tuyo. Normalmente cuando es un programa que haces partiendo de nada, esta medida es cero
  - Added: nuevo código añadido al base
  - Modified: parte del código base que ha sido cambiado
  - Deleted: parte del código base que se borra y no se utiliza para la próxima versión
  - Reused: se computan como reutilizadas aquellas líneas de código pertenecientes a otra librería, programa, etc., sólo en el caso de que ese código que reutilizas no se haya modificado. Si se modifica ya no se contabiliza como reused sino como base y los cambios se contabilizan como modificaciones (modified)

Estándar para cálculo de líneas de código

- Los elementos básicos del conteo de líneas de código son los siguientes:
  - Added and modified: Cuando el esfuerzo de añadir y el de modificar son similares conviene combinarlos, para ello se usa esta sección (es la suma de los dos, añadidos y modificados)
    - En el formulario de PSP0.1 se utiliza como campo para predecir el tamaño en locs del programa en cuestión.
  - New reusable: está dedicado a código nuevo que se ha desarrollado para incluirlo en una librería de elementos a reutilizar. Si haces código para ser reutilizado, marca este campo con un asterisco.
  - Total: medida total del programa

Estándar para cálculo de líneas de código

- Ejemplo:

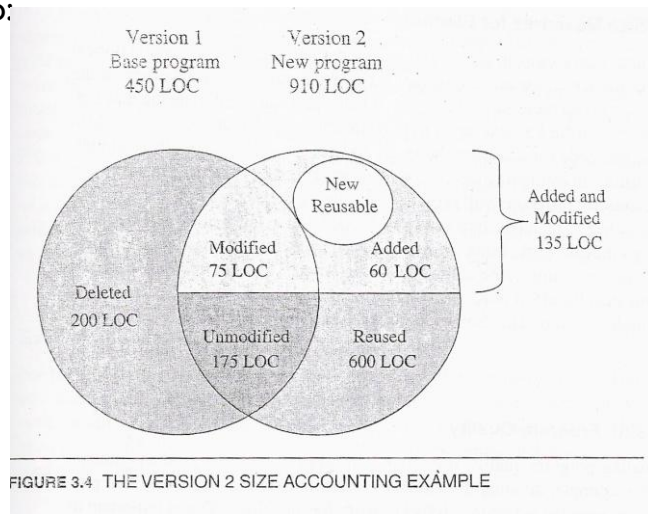


FIGURE 3.4 THE VERSION 2 SIZE ACCOUNTING EXAMPLE

Productividad

- Para calcular la productividad divide el tamaño del programa producido entre las horas que has invertido en producirlo.
- Ello te dará un volumen de trabajo realizado por hora.
- Por ejemplo si has desarrollado 600 LOC en 6 horas tienes una productividad de 10LOC por hora.

Medición

- Medimos para:
  - Entender y gestionar el cambio
  - Predecir o planificar el futuro
  - Comparar un producto, proceso u organización con otros
  - Determinar as adherencias a estándares
  - Proporcionar una base para el control
- Mediciones PSP
  - Tamaño del programa
  - Tiempo gastado por fase
  - Defectos encontrados e introducidos por fase

Medidas de tamaño

- El primer requisito para elegir una medida del tamaño del software es que sea:
  - Precisa: una medida precisa da un valor preciso para un producto. Cada vez que el tamaño sea medido el resultado debe ser el mismo, sin que quepa la opinión o subjetividad.
  - Automática (en el conteo): el conteo manual es tedioso, debe ser automatizable dicho cálculo.
  - Y específica: por ejemplo la medida del tamaño para productos escritos en C++ será diferente de la de productos escritos en Pascal, Java, etc.

Medidas de tamaño

- Por ejemplo el número de páginas podría ser una unidad de medida, pero no es precisa porque hay páginas a simple y doble espacio, en dos columnas, etc.
- De modo que cuando elijáis una unidad de medida cercioraros de que cumple con las características que os hemos indicado.

Uso de las medidas de tamaño

- El tamaño de un programa es proporcional al tiempo requerido en desarrollarlo.
- En este caso se dice que el tamaño y el tiempo están correlacionados
- La correlación es el grado en que dos grupos de datos están relacionados.
- El grado de correlación, varía entre (-1.0 – 1.0).
- Cuanto mas cerca de 1.0, mas correlacionadas están las dos variables que representan los conjuntos de datos.
- Para que un valor de correlación sea útil para estimación y planificación en PSP debe ser mayor que 0.7

Uso de las medidas de tamaño

- También es importante hablar del nivel de significancia
- La significancia mide la probabilidad con la que esa relación entre dos variables pudiera ocurrir por casualidad.
- Una significancia de 0,25 significa que un cuarto de las veces la relación entre esas dos variables se ha dado por culpa de datos aleatorios en las variables consideradas.
- Una significancia de 0,005 significa que la relación entre las dos variable consideradas se da por azar solo una vez de cada 200.
  - 0,005 es una buena significancia. 0,2 se considera pobre.

Uso de las medida de tamaño

- Encontrando la correlación y la significancia entre el tamaño y el tiempo para un número de programas, puedes determinar si el tamaño pudiera ser un buen predictor del tiempo que tardaras en realizar un desarrollo.

Uso de las medidas de tamaño

BOX 3.1 CALCULATING THE CORRELATION COEFFICIENT

1. To determine if two sets of data, x and y, are sufficiently related for planning purposes, calculate their correlation coefficient and significance.
2. Start with n pairs of numbers x_i and y_i where x is the set of estimates and y is the actual data for those estimates.
3. The correlation r of the two sets of data is given by the following equation:

$$r(x,y) = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$

4. The values of r vary from –1.0 to +1.0. If $|r| >= 0.7$, the relationship is considered good enough for estimation purposes.
5. The closer $|r|$ is to 1.0, the better the predictive value of the relationship. However, because small values of n can result in high correlations, the significance of the correlation should also be considered. The significance calculation is shown in Box 3.2.

Uso de las medidas de tamaño

BOX 3.2 CALCULATING SIGNIFICANCE

1. The significance value indicates whether the correlation of two sets of data, x and y, is sufficiently strong that the data can be used for estimating purposes or the relationship likely occurred by chance.
2. Start with n pairs of numbers x_i and y_i where x is the set of estimates and y is the actual measured values for the estimated items.
3. If the correlation $|r|$ of these data is greater than 0.7, calculate its significance.
4. Calculate the t value for this relationship as follows:

$$t = \frac{|r(x, y)|\sqrt{n-2}}{\sqrt{1-r(x, y)^2}}$$

5. Since r can be plus or minus, the absolute value, $|r(x, y)|$, is used.
6. Look up the value of the t-distribution in Table 3.1. Use the 95% column and, for $n = 5$, use 3 degrees of freedom ($d = 3$).
7. A t value of 3.182 or greater indicates a significance of less than (better than) 0.05, or less than a 5% chance of having occurred by coincidence. Values less than 0.05 are considered adequate.

Estándar para cálculo de líneas de código

- Una métrica que cumple con los requisitos que hemos impuesto para que una métrica sea fiable, es el tamaño de líneas de código fuente.
- LOC = SLOC = Source lines of code
- La correlación entre LOC y tiempo de desarrollo está demostrado que es buena, será la que usemos.
- Pero recordad que PSP trabaja bien también con otras métricas que no sean LOC y tiempo.
- El primer paso para establecer un estándar de cálculo de líneas de código es determinar la estrategia de conteo.

Estándar para cálculo de líneas de código

- If A > B
  - Then begin A:=A-B end
  - Else begin A:=A+B end;

- If A>B
  - Then

```
Begin
   A:=A-B
End
Else
  Begin
   A:=A+B
End;
```

Estándar para cálculo de líneas de código

- Vamos a seguir la siguiente guía:
  - Cuenta cada sentencia lógica, como por ejemplo cada punto y coma y las palabras clave seleccionadas
  - Escribe cada línea contable en una línea separada
  - Cuenta todas las líneas excepto las líneas en blanco y líneas comentadas
  - Cuenta cada lenguaje por separado
- Estas guías permiten automatizar en conteo de líneas de código.
- Líneas en blanco o comentadas no se cuentan porque el esfuerzo realmente está representado por las sentencias ejecutables.

Estándar para cálculo de líneas de código

- Pero cuidado!!
  - Si realmente tu consideras que el tiempo dedicado a escribir comentarios es suficientemente significativo, entonces puedes decidir contar también las líneas de comentario también.
- En este curso, por unificar, no contaremos ni las líneas en blanco ni las comentadas

Estándar de Conteo de LOC para Java

Tipo de Sentencia	**Contar**
Ejecutable	Si – Una por cada línea
No-ejecutable:	
• Declaraciones	Si– Una línea por cada declaración
• Directivas de Compilación	Si – Una línea por cada directiva
• Comentarios:	
◦ Líneas propias	No
◦ Con código fuente	No
• Líneas en blanco	No

Estándar de Conteo de LOC para Java

Estructuras	Reglas
Sentencias de selección (*if*, *else if*, else, "*?*" operator, *try* , *catch*, *switch*)	Contar una línea por cada ocurrencia. Las sentencias anidadas son contadas de una manera similar.
Sentencias de iteración (*for, while, do..while*)	**Contar una línea por cada ocurrencia.** La inicialización, condición e incremento dentro de la estructura "*for*" no son contadas. *for (i= 0; i < 5; i++)…* Cualquier expresión adicional dentro de la estructura "*for*" no es contada. *for (i = 0, j = 5; i < 5, j > 0; i++, j--)…*
Sentencias de salto (*Return, break, exit, continue, throw*)	Contar una por cada ocurrencia.
Sentencias de expresiones (llamadas a funciones, asignaciones y sentencias vacías)	Contar una línea por cada ocurrencia. Las sentencias vacías no afectan la lógica del programa y usualmente sirven como marcadores de posición para consumir CPU para propósitos de manejo de tiempos.

Estándar de Conteo de LOC para Java

Estructuras	Reglas
Sentencias en general (sentencias que finalizan con punto y coma)	Contar una línea por cada ocurrencia. Los puntos y comas dentro de sentencias "for" no son contados.
Delimitadores de bloques {,}	Cuente una línea por cada par de {,} Las llaves de inicio ({) y cierre (}) utilizados dentro de sentencias de iteración y selección no son contados. La definición de funciones, métodos y procedimientos se cuentan ya que son seguidas por un conjunto de {,}.
Directivas de compilación	Contar una por cada ocurrencia.
Declaraciones de datos	Contar una línea por cada ocurrencia. Incluye funciones, declaraciones de variables y clases.

Ejemplo 1 de conteo de LOC

```
/**
 * Este es un ejemplo de creación de un array en este caso de dos dimensiones
 */
package ejemplos;

public class matrices {
    public static void main( String args[] ) {

    // Declaramos un array de dos dimensiones con un tamaño de 3 en la
    // Decladramos el array con un tamano de 3 en su primera dimensión para
    // posteriormente declarar la segunda dimensión.

    int matriz[][] = new int[3][];
    matriz[0] = new int[2];
    matriz[1] = new int[3];
    matriz[2] = new int[4];

     // Ponemos datos en el array

    for ( int i=0; i < 3; i++ ) {
       for ( int j=0; j < matriz[i].length; j++ )
           matriz[i][j] = i * j;
    }

    // y vemos su contenido, utilizando un bucle for
    for ( int i=0; i < 3; i++ ) {
        for ( int j=0; j < matriz[i].length; j++ )
            System.out.print( matriz[i][j] );
         System.out.println();
    }

    // Intetamos acceder a un elemento que esta fuera de los limites del array
    System.out.println( "Elemento fuera de limites del array" );
    matriz[4][0] = 7;

    // El compilador lanzara una excepción de tipo ArrayIndexOutOfBounds
    }
}
```

Total LOC: 18

Ejemplo 2 de conteo de LOC

```
public class Alumno
{
   private String nombre;
   private String apellidos;
   private int añoNacimiento;
   private String grupo;

   //Si el nombre o los apellidos son cadenas null
   //lanzamos una excepción y el objeto no se crea
   public Alumno (String nombre, String apellidos, int añoNacimiento) throws Exception {
      if (nombre == null || apellidos == null)
         throw new Exception("Argumentos no válidos");
      if (añoNacimiento < 0)
         throw new Exception("Año incorrecto");
      this.nombre = nombre;
      this.apellidos = apellidos;
      this.añoNacimiento = añoNacimiento;
   }

   public String dameGrupo() {
      return grupo;
   }

   public void imprime() {
      System.out.println("nombre:" + nombre);
      System.out.println("apellidos:" + nombre);
      System.out.println("año de nacimiento:" + añoNacimiento);
      System.out.println("grupo:" + grupo);
   }
}
```

Total LOC: 25