PEC 1

Empezando…

Universitat Oberta de Catalunya

Información relevante:

- Fecha límite de entrega: 4 de octubre.
- Peso en la nota de FC: 10%.

Contenido

Información docente

Esta PEC está vinculada con el módulo teórico “Introducción al paradigma de la programación orientada a objetos” de los apuntes de la asignatura.

Prerrequisitos

Para hacer esta PEC necesitas saber:

- Conceptos básicos de algorítmica (aprendido en la asignatura anterior).
    - Conocer el paradigma de la programación estructurada.
    - La sintaxis de un lenguaje imperativo de uso común, p.ej. C, Python, etc.
    - Declaración y uso de variables de tipo primitivo (`int`, `float`, etc.).
    - Bucles (`while`, `for` y `do-while`).
    - Condicionales (`if-else` y `switch`).
    - Uso de *arrays*.
    - Creación y uso de funciones (en nuestro contexto llamados “métodos”).

Objetivos

Con esta PEC el equipo docente de la asignatura busca que:

- Uses Java como un lenguaje de programación imperativo y así conozcas la sintaxis básica de este lenguaje.
- Te familiarices con la dinámica de la asignatura: metodología, tipología de enunciados, nomenclaturas, etc.

Resultados de aprendizaje

Con esta PEC debes demostrar que eres capaz de:

- Configurar un entorno de trabajo básico para programar con Java: JDK y Notepad++.
- Implementar programas sencillos en Java basados en el paradigma de la programación estructurada.
- Entender qué hace un programa o código dado por un tercero.
- Configurar y usar con cierta soltura un entorno de desarrollo integrado (IDE) como Eclipse.
- Ser capaz de depurar (*debug*) en un IDE un programa con errores para corregirlo.
- Utilizar test unitarios para determinar que un programa es correcto.

Enunciado

Esta PEC contiene 3 tareas (actividades no evaluables) y 3 ejercicios (sí evaluables). Debes entregar tu solución de los 3 ejercicios evaluables (ver el último apartado).

Debido a que las actividades están encadenadas (i.e. para hacer una se debe haber comprendido la anterior), **es altamente recomendable hacer las tareas y ejercicios en el orden en que aparecen en este enunciado**.

Tarea 1 – Configuración del entorno básico

Antes de empezar debes:

- Instalar el JDK en tu ordenador. Para ello lee el apartado "1.1. Instalando JDK" de la Guía de Java. Además se recomienda leer el apartado 1.2 de la Guía de Java.
- Usar un editor de texto plano (p.ej. el que viene con tu sistema operativo): vim, notepad, TextEdit, etc. No obstante, para Windows recomendamos **Notepad++**.

Una vez realizados los pasos anteriores, puedes hacer la siguiente tarea para comprobar que el entorno básico para desarrollar programas en Java lo tienes bien instalado. Para ello, sigue estos pasos:

1. Abre el directorio `PAC1Task1` que te proporcionamos con el enunciado.
2. Allí encontrarás un fichero llamado `PAC1Task1.java`.
3. Abre un terminal o procesador de comandos (cmd).
4. En el terminal ubícate en el directorio `PAC1Task1` y una vez en él ejecuta el comando:

```
javac PAC1Task1.java
```

5. Si todo va bien, en el terminal no sucederá absolutamente nada. Esto quiere decir que el fichero `PAC1Task1.java` no ha producido ningún error. Sin embargo, si miras dentro del directorio `PAC1Task1`, debería haberte aparecido un nuevo fichero llamado `PAC1Task1.class`. Este nuevo fichero es el *bytecode* de `PAC1Task1.java`.
6. A continuación vamos a ejecutar `PAC1Task1.class`. Para ello, ejecuta el siguiente comando en el terminal que tienes abierto (que está ubicado en el directorio `PAC1Task1`):

```
java PAC1Task1
```

7. Verás que el programa escribe el siguiente mensaje: `Draw`
8. Ahora ejecuta el programa `PAC1Task1` pasándole argumentos. Por ejemplo:

```
java PAC1Task1 1 2 3 4 5
```

9. La ejecución anterior debería producir la siguiente salida por pantalla: `Odd`

10. Haz más pruebas pasándole diferentes argumentos. Después abre el fichero `PAC1Task1.java` y observa el código que contiene. Asimismo prueba otras ejecuciones del programa. ¿Qué crees que hace este programa? Por ejemplo, calcular la media de los números pasados como argumentos.

11. Respuesta:

> Este programa indica por pantalla si en el listado de número
> sfacilitado hay más números pares que impares (salida even),
> más números impares que pares (salida odd) o igual cantidad
> (salida draw).

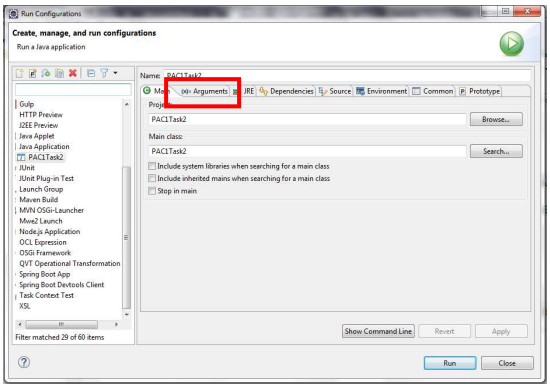
Tarea 2 – Configurando Eclipse

Antes de continuar debes:

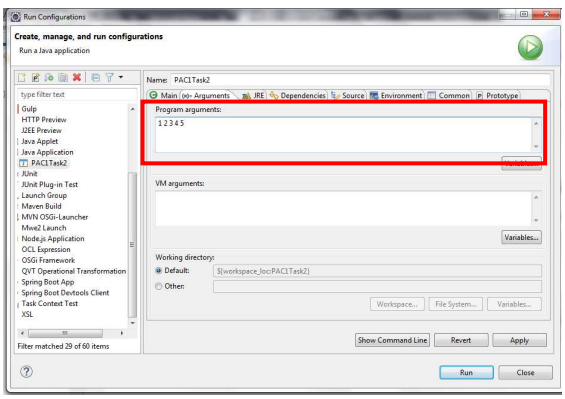
- Instalar y configurar el IDE Eclipse en tu ordenador. Para ello lee el apartado “1.3. Eclipse IDE” de la Guía de Java.

Una vez realizado el paso anterior, **abre o importa el proyecto `PAC1Task2` con Eclipse**. En él verás un proyecto con un fichero llamado `PAC1Task2.java`. Una vez abierto el proyecto, haz click en el botón `Run` . Verás que el resultado es el mismo que en la tarea anterior, puesto que el código es el mismo. La única diferencia es que ahora pertenece a un proyecto Eclipse. Si siempre le das a `Run`, siempre te aparecerá `"Draw"`. Normal, porque no le estás pasando argumentos. ¿Cómo pasarle argumentos a la ejecución?

1. Ves al icono de `Run` y al lado verás una flecha que despliega un menú.
2. En él haz click en la opción `"Run configurations…"`.
3. Se abrirá una ventana como la que se te muestra a continuación.



4. Ves a la pestaña `Arguments` y en él escribe unos números separados por espacios en el apartado `Program arguments`.



5. Si ahora haces click en el botón `Run` de la ventana o, posteriormente, en el botón `Run`, se ejecutará con los valores que has escrito como argumentos en `Program arguments`. Si quieres modificarlos, debes ir a `Run configurations`…

Ejercicio 1 – Debugging (2 puntos)

Para hacer este ejercicio **abre o importa el proyecto `PAC1Ex1` con Eclipse**. Verás que aparecen errores. Los errores que Eclipse muestra, o bien son sintácticos, o bien hacen referencia a elementos que Eclipse no conoce (estos errores también entran dentro del apartado sintáctico porque para Eclipse son como si estuvieran mal escritos).

a) Corrige los errores sintácticos del programa de manera que no contenga ninguno.

(1 punto)

Ahora si ejecutas el programa verás que el resultado no es el esperado. ¿Por qué? Pues porque contiene errores de lógica, es decir, la funcionalidad no está bien programada. Evidentemente Eclipse no es mágico y no sabe qué pretendes programar. Así pues debes comprobar mediante "trazas" (i.e. ejecutar el programa paso a paso) para saber qué está haciendo el programa y detectar dónde está el o los problema/s. Para ello puedes usar la funcionalidad de debugación (en español, depuración) que ofrece Eclipse.

b) Corrige los errores lógicos del programa de manera que se ejecute como se espera. Por ejemplo:

```
96.8ºF = 36.0ºC
100ºF = 37.777777ºC
32ºF = 0.0ºC
```

(1 punto)

Requisito mínimo para evaluar este ejercicio: el programa debe funcionar como es esperado, i.e. se deben realizar los apartados (a) y (b) correctamente.

Tarea 3 – Testing con JUnit

En el Ejercicio 1 la única manera que has tenido para saber si el programa se comportaba como era esperado ha sido ejecutándolo varias veces y viendo su salida por pantalla. Hacer test y detectar errores utilizando el método del propio lenguaje que escribe por pantalla (p.ej. `printf`, `print`, `println` o equivalente del lenguaje que utilices habitualmente) es una manera muy básica –muy de principiante–, aunque todo el mundo la suele utilizar en algún momento por su comodidad.

Dado que el *testing* es muy importante para garantizar que un programa es correcto y se comporta como se espera, con el tiempo ha surgido una manera de desarrollar software basada justamente en el testeo. Es lo que se conoce como TDD o *Test-Driven Development* (i.e. desarrollo conducido/dirigido por test). Esta forma de desarrollar tiene como lema: *"un programador debe escribir los casos de test antes que el código que dichos test van a comprobar"*.

A partir de ahora en esta asignatura escribirás o te proporcionaremos test para todos los ejercicios que requieren codificación. Para ello usaremos la librería JUnit. Es por este motivo que antes de continuar debes:

- Leer el apartado "5.3. JUnit" de la Guía de Java que explica cómo añadir, configurar y usar JUnit en un proyecto Java en el IDE Eclipse.

Ejercicio 2 (4 puntos)

Abre o importa el proyecto `PAC1Ex2` en Eclipse. Debes codificar los `TODO` (del inglés, *"to do"*, en español, *"por hacer"*, o usando una traducción más libre: *"pendiente"*). Este ejercicio básicamente consiste en codificar los siguientes 2 métodos/funciones:

- `incomeTaxPayable`: recibe el salario anual de un trabajador y calcula los impuestos anuales que debe pagar al Estado según la tabla de tramos siguiente:

Tramos	Porcentaje/Tasa (%)
Primeros 20000 €	0% (los primeros 20000€ no tributan)
Siguientes 20000 €	10%
Siguientes 20000 €	20%
El resto	30%

  Así pues, una persona que cobra anualmente 85432€ debe pagar al Estado:

  20000€ * 0% + 20000 * 10% + 20000 * 20% + 25432 * 30% = 13629.60 €

  (2 puntos: 1 pt. test proporcionados; 0.5 pts. test extras; 0.5 pts. otros)

- `pensionContribution`: recibe el salario mensual de un trabajador y calcula su cotización mensual a la Seguridad Social en concepto de pensión. La fórmula para dicho cálculo es la siguiente: se mira cuántas veces el salario mensual del trabajador es múltiplo de `200`. Dicha cantidad, que llamaremos *n*, se redondea a la baja (es decir, si el salario mensual es `4.86` veces `200`, dicho valor se simplificará a `4` veces). Una vez determinado el valor de *n*, se calculará la siguiente fórmula *n* veces: `salario * tasa`; donde el valor inicial de la tasa es `0` y dicho valor se incrementará `0.01` en cada iteración. Los valores de las *n* iteraciones se sumarán, dando como resultado final la contribución mensual del trabajador en concepto de pensión. Veamos un ejemplo:

  1428.57 €/mensuales → n = 1428.57/200 = 7.14 = 7

n = 0	1428.57 * 0	0 €
n = 1	1428.57 * 0.01	14.28 €
n = 2	1428.57 * 0.02	28.57 €
n = 3	1428.57 * 0.03	42.85 €
n = 4	1428.57 * 0.04	57.14 €
n = 5	1428.57 * 0.05	71.42 €
n = 6	1428.57 * 0.06	85.71 €
Total (suma)		299.97 €*

  * El cálculo en este ejemplo se ha redondeado para no arrastrar demasiados decimales. El resultado exacto que debe dar el programa es 300 €.

  (2 puntos: 1 pt. test proporcionados; 0.5 pts. test extras; 0.5 pts. otros)

Requisito mínimo para evaluar este ejercicio: el programa debe pasar todos los test proporcionados para el método `incomeTaxPayable`.

Ejercicio 3 (4 puntos)

Abre o importa el proyecto `PAC1Ex3` en Eclipse. Debes codificar los `TODO` (del inglés, *"to do"*, en español, *"por hacer"*, o usando una traducción más libre: *"pendiente"*). Básicamente:

- `lengthSubVector`: recibe un *array* de enteros (`int`) y un `boolean` que indica si debe contar números pares (`true`) o impares (`false`). Devuelve el número de casillas del *array* que contienen un número par (si el valor del parámetro de tipo `boolean` es `true`) o impar (si el valor del parámetro de tipo `boolean` es `false`).

  `lengthSubVector({1,2,3},true)` → `1`

  `lengthSubVector({1,2,3},false)` → `2`

  Si el *array*/vector pasado como argumento:

  - Está vacío, entonces el valor devuelto debe ser `0`.
  - Es el valor `null`, entonces debe imprimir por pantalla el mensaje `"[ERROR] Vector is null!!"` y devolver un *array* vacío.

  (3 puntos: 1 pt. test proporcionados; 1 pt. test extras; 1 pt. otros)

- `subVector`: devuelve un *array* cuyas casillas son los números pares o impares del *array* pasado como parámetro. Para determinar qué tipo de números incluye el *array* devuelto (i.e. pares o impares), este método debe mirar el tipo del primer elemento del *array*. Así pues, si el primer elemento es par, entonces el *array* devuelto por `subVector` contendrá los números pares del *array* pasado como parámetros. En caso de que sea impar, entonces contendrá los valores impares. La longitud del *array* devuelto debe coincidir con el número de elementos que satisfacen la condición par o impar, según sea el caso (es decir, no se puede devolver un *array* con la misma longitud del *array* original pero con las casillas que no satisfacen la condición con valor `null`).

  `subVector({1,2,3})` → `{1,3}` (devuelve los números impares porque `"1"` es impar)

  `subVector({2,4,3})` → `{2,4}` (devuelve los números pares porque `"2"` es par)

  Si el *array*/vector pasado como argumento:

  - Está vacío, entonces el valor devuelto debe ser el *array* original (i.e. un *array* vacío).
  - Es el valor `null`, entonces debe imprimir por pantalla el mensaje `"[ERROR] Vector is null!!"` y devolver `null`.

  (1 punto: 1 pt. test proporcionados)

Requisito mínimo para evaluar este ejercicio: el programa debe pasar todos los test proporcionados para el método `lengthSubVector`.

Formato y fecha de entrega

Tienes que entregar un fichero *.zip, cuyo nombre tiene que seguir este patrón: `loginUOC_PEC1.zip`. Por ejemplo: `dgarciaso_PEC1.zip`. Este fichero comprimido tiene que incluir los siguientes elementos:

- El proyecto de Eclipse `PAC1Ex1` completado siguiendo las peticiones y especificaciones del Ejercicio 1.
- El proyecto de Eclipse `PAC1Ex2` completado siguiendo las peticiones y especificaciones del Ejercicio 2.
- El proyecto de Eclipse `PAC1Ex3` creado por ti con el código que cumple las especificaciones del Ejercicio 3.

El último día para entregar esta PEC es el **4 de octubre de 2020** antes de las 23:59. Cualquier PEC entregada más tarde será considerada como no presentada.