SESIÓN 02: Estructuras de control: Selectivas o Condicionales

Objetivos

- Introducir el uso de los repositorios como herramienta para el control de versiones de código.
- Saber construir y utilizar una estructura selectiva simple, doble y múltiple.
- Implementar algoritmos sencillos.

CONEXIÓN Y USO DE RESPOSITORIOS

Utilización del entorno de programación de **ECLIPSE**. El alumno debe crear un **proyecto Java**, donde se realizarán todas las sesiones de la asignatura y conectarlo con su repositorio personal donde subirán todos los ejercicios resueltos.

El **nombre del proyecto Java del alumno** será:
apellidos y nombre del alumno seguido de **IP2022**, ej: **CorralLiriaAntonioLeopoldoIP2022**

Tal y como se verá en el archivo **GuiaRapidaRepositorios.pdf**, cada alumno tiene asignado un **repositorio personal** con un formato de **apellidosnombreip2022**, por ejemplo, para Antonio Leopoldo Corral Liria tenemos el respositorio **corralliriaantonioleopoldoip2022**, es decir, los **apellidos** y **nombre** en minúscula seguidos de **ip2022**, sin espacios.

Es necesario que el proyecto Java del alumno se suba (**Commit**) al repositorio personal después de cada sesión de prácticas. Estos son los pasos para "conectar" el proyecto java del alumno con su repositorio personal:

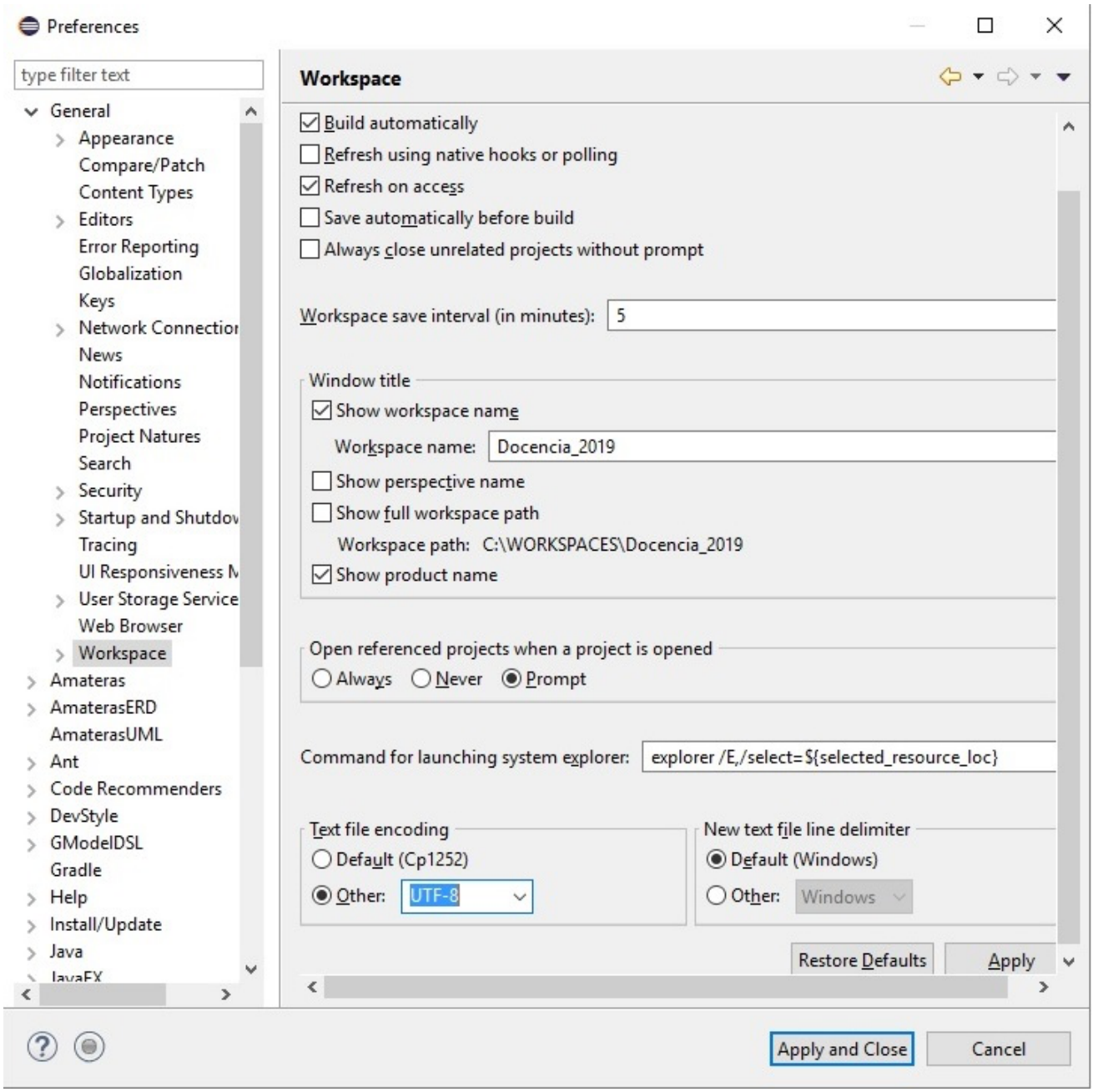
1. En primer lugar tenemos que crear el enlace al repositorio personal desde la ***perspectiva SVN*** (si aún no lo hemos hecho).
2. Después creamos el proyecto Java (si no se ha hecho) para esta asignatura en nuestro *workspace* (***perspectiva Java***) con un nombre similar a **CorralLiriaAntonioLeopoldoIP2022**
3. Por último, "conectamos" nuestro proyecto Java con el repositorio personal de este modo:

En la ***perspectiva Java***, sobre el nombre del proyecto de la asignatura, hacemos clic con el botón derecho y seleccionamos: **Team > Share Project… > SVN** (Next) >
y lo vinculamos al enlace a nuestro repositorio personal, que en nuestro ejemplo es
http://svndoc.ual.es:9090/svn/corralliriaantonioleopoldoip2022
pulsamos (Next) > y seleccionamos **Advanced Mode** (Next) > **Finish**.

Es muy importante que se seleccione **Advanced Mode**, pues de no hacerlo así el proyecto no colgaría del *trunk* y entonces no estaría bien enlazado para su posterior corrección.

Nota importante: Como ya se advirtió en la sesión 01 y en clase de teoría, no hay que escribir nunca caracteres acentuados ni la *ñ* (tanto en su versión minúscula como en mayúscula), ni incluso en los comentarios. Para evitar problemas relacionados con esta restricción es necesario realizar la siguiente modificación en la configuración de nuestro Eclipse.

Windows > Preferences > General > Workspace
Y entonces seleccionar en el cuadro Text file encoding en la opción Other **UTF-8** y pulsar el botón **Apply and Close**, tal y como se indica en la siguiente figura



Ejercicios Propuestos

Nota importante: Siga el esquema de nombrado de paquetes que se indicó en la sesión 01 es decir: **org.ip.sesion02**. En ese paquete se crearán todos los programas Java que se proponen en esta sesión dándoles un nombre apropiado en función de lo que realiza el programa en cuestión y que se indica en cada ejercicio entre paréntesis y en negrita **(NombrePrograma)**.
Al final de la sesión, el alumno deberá cargar (o subir) el trabajo realizado a su repositorio indicando la clave correspondiente a la sesión.

1. En un conocido centro comercial de la ciudad se han detectado unos defectos en aquellos artículo cuyos códigos se encuentran entre los aquí expuestos:

   Del 14681 al 15681
   Del 70001 al 79999
   Del 99999 al 110110

   Implemente un programa **(CodigoDefectuoso)** en Java que a partir del código de un artículo, muestre un mensaje indicado si el artículo es o no defectuoso.

   Ejemplo de ejecución:

   ```
   El codigo 25000 corresponde a un articulo NO DEFECTUOSO
   ```

2. Desarrolle un programa en Java **(OperacionAritmetica)** que a partir de dos números y un símbolo operador (* ⇒ Multiplicar, / ⇒ Dividir, + ⇒ Sumar, - ⇒ Restar), realice la operación con los números anteriores siempre que sea posible.

   Ejemplo de ejecución:

   ```
   Los datos son:
   x = 7, y = 2
   La operacion elegida es: /

   El resultado de la operacion es: 7 / 2 = 3.5
   ```

3. Implemente un programa en Java **(EcuacionSegundoGrado)** que muestre por pantalla las raíces de una ecuación de 2º grado dados los valores de a, b y c. a$x^2 + bx + c = 0$.

   Ejemplo de ejecución:

   ```
   SOLUCION DE UNA ECUACION DE SEGUNDO GRADO

   Valores de los coeficientes

   a = 1, b = 2, c = 1

   Una unica raiz de valor doble
   x = -1.0
   ```

Pruebe el programa implementado con los valores que se indican en la siguiente tabla, debiendo generar la salida que se muestra en cada caso

a b c	Salida a consola
0 1 1	SOLUCION DE UNA ECUACION DE SEGUNDO GRADO Valores de los coeficientes a = 0, b = 1, c = 1 No es una ecuacion de segundo grado
2 1 1	SOLUCION DE UNA ECUACION DE SEGUNDO GRADO Valores de los coeficientes a = 2, b = 1, c = 1 No tiene solucion real
1 -3 2	SOLUCION DE UNA ECUACION DE SEGUNDO GRADO Valores de los coeficientes a = 1, b = -3, c = 2 Dos raices de valores x1 = 2.0 x2 = 1.0

Trabajo Autónomo

4. En este ejercicio se implementará un programa en Java que compruebe si una fecha es correcta **(FechaCorrecta)**. Para ello se proporcionarán valores a las variables *día*, *mes* y *anio*, y serán éstas las que se comprueben. Destacar que se deben considerar años bisiestos (febrero puede tener 29 días para este caso) y que el número correcto de años va desde 1500 hasta 3000.
Recordar que un año es bisiesto si: *anio % 4 = 0* **y** *anio % 100 ≠ 0* **o** *anio % 400 = 0*.

Ejemplo de ejecución:

```
Fecha correcta: 29/2/2020
Fecha incorrecta - 29/2/2019
Fecha incorrecta - 31/11/2019
Fecha incorrecta - 31/12/1019
Fecha incorrecta - 31/13/2019
```

5. A partir de un *mes* (**m**), *día* (**d**), y *año* (**y**), implemente un programa en Java **(DiaSemana)** que muestre el día de la semana según el Calendario Gregoriano, **d0**. Para los meses utilizar: 1 para enero, 2 para febrero, 3 para marzo, y así sucesivamente. La salida por pantalla, **d0**, es 0 para el Domingo, 1 para el Lunes, 2 para el Martes, 3 para el Miércoles, y así sucesivamente. Utilice las siguientes expresiones (fórmulas) para obtener el valor de **d0**.

$$y0 = y - (14 - m) / 12$$
$$x = y0 + y0 / 4 - y0 / 100 + y0 / 400$$
$$m0 = m + 12 * ((14 - m) / 12) - 2$$
$$d0 = (d + x + (31 * m0) / 12) \% 7$$

Ejemplo de ejecución:

```
El dia de la semana correspondiente al 3/3/2006 es:

VIERNES
```

6. La tarifa que aplica un taxista es la siguiente:

- Una cantidad fija de 18 euros si no se sobrepasan los 30 Km.
- Para más de 30 Km, se consideran los siguientes supuestos:
  - ✓ Si no se sobrepasan los 100 Km, 0.85 euros por Km que exceda de los 30, además de los 18 euros.
  - ✓ Si sobrepasa los 100 Km, 0.65 euros por Km que exceda de los 100, 0.85 euros por Km desde los 30 a los 100 y los 18 euros.

Desarrolle un programa en Java que, a partir de los kilómetros recorridos, calcule y muestre por pantalla el total a pagar según la tarifa anterior. **(TarifaTaxi)**

Ejemplo de ejecución:

```
CALCULO TARIFA TAXI

Kilometros recorridos => 101
El importe total a pagar es 78,15 euros
```

7. Implemente un programa en Java de nombre **(CambioMonedas)**, que dado un valor real que indica una cantidad de euros, por ejemplo 21.38 euros, muestre por pantalla la cantidad de monedas de curso legal (2 euros, 1 euro, 50 céntimos, 20 céntimos, 10 céntimos, 5 céntimos, 2 céntimos y 1 céntimo) de cada tipo en que se debe de devolver dicha cantidad de euros.

Ejemplo de ejecución:

```
Para devolver 21.38 euros en monedas, hay que dar:
10 monedas de 2 euros
1 monedas de 1 euro
1 monedas de 20 centimos
1 monedas de 10 centimos
1 monedas de 5 centimos
1 monedas de 2 centimos
1 monedas de 1 centimo
```

8. Desarrolle un programa en Java de nombre **(SegundoSiguiente)**, que dada un instante de tiempo en formato **hora:minuto:segundo**, muestre el instante de tiempo un segundo después. Por ejemplo, si el instante de tiempo actual es 11:42:59, el instante de tiempo un segundo más tarde es 11:43:00. Asumimos que los valores de **hora** (0 ≤ hora ≤ 23) de **minuto** (0 ≤ minuto ≤ 59) y de **segundo** (0 ≤ segundo ≤ 59) se introducen correctamente.

Ejemplo de ejecución:

```
Instante de tiempo actual             : 03:29:36
Instante de tiempo un segundo despues : 03:29:37

Instante de tiempo actual             : 11:42:59
Instante de tiempo un segundo despues : 11:43:00

Instante de tiempo actual             : 11:59:59
Instante de tiempo un segundo despues : 12:00:00

Instante de tiempo actual             : 23:59:59
Instante de tiempo un segundo despues : 00:00:00
```