Anexo UD02: Herramientas útiles del IDE

1. Herramientas útiles del IDE

1.1. Netbeans

- Code templates.
    - `psvm + TAB` para `public static void main()`
    - `sout + TAB` para `System.out.println("")`
    - Consultar más o personalitzar: `Tools/Options/Editor/Code Templates`
- Fix imports (botón derecho `Fix imports`)
- Fix package (cuando el archivo no está en el paquete correcto, podemos modificar el código o la estructura de carpetas)
- Si una palabra aparece subrayada en rojo, podemos pulsar ALT+ENTER para ver las sugerencias, por ejemplo:
    - Renombrar el archivo según la clase
- Refactorizar nombres de variables, métodos, clases... (Cambiamos el nombre de una variable y el modifica todas las apariciones en el código fuente. Lo hace de una manera más inteligente que el "Reemplazar todo")
- CONTROL+ESPACIO ver parámetros de métodos y/o completar nombres.
- ALT+SHIFT+F para formatear el código
- Insert Code:
    - ALT+INSERT o botón derecho `Insert Code...`
        - Constructor
        - Getters
        - Setters
        - Getters and setters
        - ...
- ```
  1 Date fecha = new ~~Date~~ (2021, 2, 19); //Date constructor is deprecated
  ```
- Cuando el nombre de una classe aparece en cursiva es porque es estática (`static`)
- Ejecutar con parámetros: `Run/Set Project Configuration/Customize.../Run` puedes definir cual es la clase principal en el campo `Main Class:` con el formato `<paquete>.<Clase>` y los parámetros que recibirá el método `main` de dicha clase en el campo `Arguments`: separando por espacios los parámetros.
- Depuración:
    - `Debug file` (depurar archivo, CTRL+SHIFT+F5)
    - `Togle Line breakpoint` (activar/desactivar un breakpoint, cuadrado rojo, CTRL+F8, ojo en linux)
    - `Step over` (Siguiente paso, sin entrar en detalles, F8)
    - `Step into` (Siguiente paso, entrando en detalles, F7)
    - `Continue` (Continuar, F5)
    - `New Breakpoint...` (personalizar las condiciones y lugar del breakpoint CTRL+SHIFT+F8)
  ```

    - `New Watch...` (inspeccionar el contenido de la variable o expresión mientras se depura el código)
- `TODO`: Cuando nuestros programas se hacen grandes queremos dejar anotaciones de lo que nos "dejamos por hacer", para eso sirve el TODO (del inglés "por hacer"), si añadimos en un comentario las palabras @todo, TODO, FIXME (y alguna más que puedes personalizar en `Tools/Options/Team/Action Items`) te apareceran disponibles en la ventana de acciones (`Window/Action items`). Mi recomendación en Netbeans es que modifiques la plantilla para que solo reconozca cuando escribas "TODO:" (con dos puntos), y así evitar confusiones con palabras usuales en español como "méTODO".

1.2. Eclipse

- Code templates.
    - `main + CONTROL+ESPACIO` y seleccionar `main method` para `public static void main()`
    - `sysou + CONTROL+ESPACIO` para `System.out.println("")`
    - Consultar más o personalitzar: `Window/Preferences/Java/Editor/Templates`
- Organize imports (botón derecho `Source/Organize imports` o `CONTROL+SHIFT+O`)
- Fix package (cuando el archivo no está en el paquete correcto, podemos modificar el código o la estructura de carpetas)
- Si una palabra aparece subrayada en rojo, podemos pulsar F2 para ver las sugerencias, por ejemplo:
    - Renombrar el archivo según la clase
- Refactorizar nombres de variables, métodos, clases... (Cambiamos el nombre de una variable y el modifica todas las apariciones en el código fuente. Lo hace de una manera más inteligente que el "Reemplazar todo")
- El completado de parámetros de métodos y/o completar nombres aparece automáticamente, una vez elegimos el método deseado podemos alternar entre los parámetros con la tecla `TAB`.
- CONTROL+SHIFT+F para formatear el código
- Generar código automáticamente:
    - Dentro del menú `Source`
        - Generate Getters and Setters...
        - Generate toString()...
        - Generate Constructor using fields...
        - Generate Constructor from Superclass...
- Cuando el nombre de una clase aparece tachado es porque es mejor no utilizarlo y hay que buscar una alternativa (`dreprecated`)

```
1 Date fecha = new ~~Date~~ (2021, 2, 19); //Date constructor is deprecated
```

- Cuando el nombre de una classe aparece en cursiva (en las invocaciones) es porque es estática (`static`)

- Ejecutar con parámetros: `Run/Run Configurations` en la parte izquierda debes escoger tu clase dentro del apartado `Java Application` y en la parte derecha puedes definir cual es la clase principal en la pestaña `Main` y los parámetros que recibirá el método `main` de dicha clase en la pestaña `Arguments` separando por espacios los parámetros.
- Depuración:
    - `Debug` (depurar archivo, F11)
    - `Togle Line breakpoint` (activar/desactivar un breakpoint, circulito azul, CTRL+SHIFT+B)
    - `Step over` (Siguiente paso, sin entrar en detalles, F6)
    - `Step into` (Siguiente paso, entrando en detalles, F5)
    - `Continue` (Continuar, F8)
    - No he encontrado la manera de condicionar un `breakpoint` tal y como lo hace Netbeans
    - Menú `Run/Watch` (inspeccionar el contenido de la variable o expresión mientras se depura el código)
- `TODO`: Cuando nuestros programas se hacen grandes queremos dejar anotaciones de lo que nos "dejamos por hacer", para eso sirve el TODO (del inglés "por hacer"), si añadimos en un comentario las palabras @todo, TODO, FIXME (y alguna más que puedes personalizar en `Window/Preferences/General/Editors/Task tags`) te aparecerán disponibles en la ventana de acciones (`Window/Show view/Tasks`). Eclipse solo reconoce el TODO que va junto al comentario es decir `//TODO` y no genera confusiones con palabras usuales en español como "méTODO". Además eclipse establece prioridades y se puede personalizar la presentación y prioridades.

2. Fuentes de información

- Wikipedia
- Code&Coke (Fernando Valdeón)
- Apuntes IES El Grao (Mª Isabel Barquilla?)
- Apuntes IOC (Marcel García)
- Apuntes José Luis Comesaña
- Apuntes IES Luis Vélez de Guevara 17-18 (José Antonio Muñoz Jiménez)