DE LA CABEZA AL CÓDIGO

Una metodología para desarrollar el pensamiento computacional

Al principio tendrás que seguir esta metodología paso a paso, pensando detenidamente cada apartado. A medida que vayas adquiriendo más práctica, te saltarás algunos pasos de forma natural, intuyendo el código tienes que escribir en cada ocasión. Eso sí, si te pierdes o te atascas en algún punto, siempre podrás volver a aplicar la metodología desde el principio hasta conseguir analizar y descomponer el problema en partes más pequeñas que sean suficientemente sencillas para ser codificadas directamente.

FASE PREVIA:

Si queremos traducir a código cierta idea que tenemos en la cabeza es completamente necesario que esa idea sea la correcta.

Si el enunciado de un problema dice "X" pero tú entiendes "Y" entonces, en el mejor de los casos, conseguirás codificar "Y" pero no estarás resolviendo el problema que pide el enunciado.

ASÍ QUE, POR FAVOR, LEE CON ATENCIÓN Y PREGUNTA LO QUE NO ENTIENDAS. EL PROFESOR/A SIEMPRE ES EL RESPONSABLE DE QUE UN ENUNCIADO SE ENTIENDA CORRECTAMENTE.

FASE ESTRUCTURAL:

Con esta fase rompemos el bloqueo mental que podemos tener al enfrentarnos al "fichero en blanco" para ello escribiremos la estructura-esqueleto de las clases que hayamos identificado en el enunciado, prestando especial atención a las propiedades y métodos. No le des código a los métodos, solo crea su estructura (cabecera + llaves + opcionalmente un return).

Cuida la indentación y respeta siempre las reglas de nombrado de los distintos elementos del lenguaje.

FASE ALGORÍTMICA:

Ahora toca analizar qué hay que hacer en cada método, realizando para ello dos tareas:

- Reflexionar sobre el punto de partida, leyendo con atención la cabecera del método y haciéndonos las siguientes preguntas:
    - ¿Qué datos nos va a proporcionar la clase que llame a este método? ¿Qué representa cada parámetro de entrada?
    - ¿Qué propiedades de la clase se necesitan para resolver el problema?
    - ¿Qué dato tenemos que construir en el método? ¿Qué representa el parámetro de salida?

- Anotar con comentarios qué secuencia de pasos tenemos que dar para resolver el problema. Aquí es importante centrarse en el QUÉ y no en el CÓMO. Por ejemplo, podríamos escribir cosas como:

```
// Pedir el precio al usuario
// Si el precio es menor que cero emitir un mensaje de error
```

En este punto NO nos interesa decir CÓMO se hacen las cosas. Es decir, no queremos cosas como:

```
// Crear un objeto Scanner
// Hacer un nextLine
// Hacer un switch con la respuesta del usuario
```

FASE DE CODIFICACIÓN:

Ahora toca darle código a cada método, centrándonos en CÓMO traducir los comentarios que hemos escrito en un código equivalente. Para ello vamos a preguntarnos lo siguiente:

- ¿Qué estructura "resuelve mejor" el paso descrito en el comentario? ¿una asignación, creación de objeto, condicional, repetitiva…? ¿Necesitamos más de una?
- ¿Nos podemos apoyar en el IDE? ¿Nos genera algún código-plantilla que podamos usar?
- ¿Nos hace falta crear alguna variable auxiliar para guardar algún dato de forma temporal o nos puede venir bien para simplificar la escritura del código?
- En el caso de que haya que retornar algún valor ¿cómo construimos el valor?

En este punto no debe haber fallos sintácticos, apóyate en las "chuletas" de la espiral correspondiente para usar la sintaxis del lenguaje correctamente.

FASE DE PRUEBAS:

Lo ideal es que cada vez que terminemos un método lo probemos, aunque otras veces lo acabemos probando todo "al final". Para ello, creamos una clase *PruebaClaseX,* le añadimos un *main* y probamos el método que hemos creado.

Si el método no funciona correctamente y sabemos dónde se produce el error, lo modificamos para arreglarlo y volvemos a probar. En el caso de que no sepamos dónde está el error entonces tenemos que DEPURAR (debug) el código para comprender qué está fallando y poder arreglarlo.

FASE DE REFACTORIZACIÓN:

Nuestro método ya está probado y funciona correctamente, pero todavía no hemos terminado. Ahora toca pensar si podemos escribir el código de otra forma "más limpia", que se lea mejor o que sea más sencilla. Podemos cambiar el nombre de variables o parámetros para que sean más descriptivos, utilizar constantes, poner las condiciones de los if en "afirmativo", cambiar una estructura por otra que sea más clara o que "encaje" mejor, "extraer" en un método a parte un trozo del código…

Ahora sí lo tienes, ve a por el siguiente método a codificar.

REFLEXIONES FINALES

Estás aprendiendo a andar. No quieras correr.

La solución de un problema no aparece por "inspiración divina" si no que EMERGE POCO A POCO a medida que te sumerges más y más en los detalles del problema. Para aprender a nadar tienes que mojarte.

Muchas veces es útil pensar en cómo resolverías un problema similar de la vida real y después intentar aplicar los mismos pasos al código.