Confiabilidad, pruebas y análisis de la información

Ejemplos nomenclatura

Servicio Nacional de Aprendizaje SENA

Variables o atributos: inician en minúscula y en singular si es uno solo, plural si son varios elementos, si es una palabra compuesta, se coloca pegada y con mayúscula la primera letra de la siguiente palabra. Ejemplo: edad, valorHora.

Constantes: no usan camel case, usan underscore case. Ejemplo: VALOR_HORA

Clases: Inician en mayúscula y se colocan en singular. Ejemplo: Persona.

Métodos: Se nombran usando camelCase, se usan las mismas reglas que para las variables

Paquetes: El nombrado de los paquetes se debe colocar en minúsculas

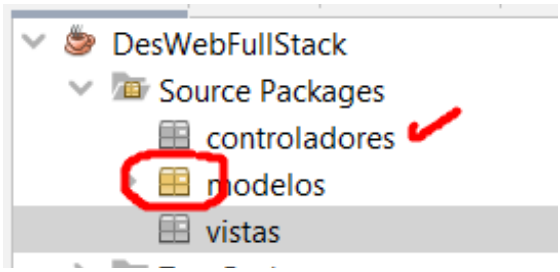
Interfaces: Para el nombrado de las interfaces, se sigue tal cual como una Clase

Cabe destacar que esta nomenclatura no es obligatoria, pero es una forma de organizar su código y trabajar con buenas prácticas de programación en Java. Dicha nomenclatura puede variar dependiendo el lenguaje de programación con el que se trabaje, no es la misma para todos.

Particularmente, al final del capítulo, construirá un proyecto con NodeJS, usando una estructura particular con el nombrado de los namespaces o carpetas, archivos, variables, métodos y demás.

Paquetes

Los paquetes (namespaces, llamados en otros lenguajes de programación), se usan también para organizar el código de tus proyectos. Normalmente se usan para colocar en ellos clases de una misma característica. A continuación, se muestra una imagen con paquetes en el IDE de NetBeans:



Los paquetes en Java se nombran con minúsculas.

Clase

A una clase se le compara como una plantilla, del cual salen varios objetos. Los objetos son instancias de una clase. Supongamos que tenemos un molde para crear figuritas de animales, las figuras serían los objetos, el molde es la clase. A continuación, se muestra el código de una clase en Java:

```
public class Persona { // nombre de la clase

    // atributos
    String nombre;
    int edad;
    char genero;
```

```
    //... métodos
}
```

Interfaces

Una interface tiene una estructura parecida a la de una clase, pero no se coloca la palabra “class” sino “interface”, se usa para declarar métodos abstractos principalmente, dichos métodos son públicos también. Los métodos abstractos no tienen cuerpo, podrás notar a continuación que termina en paréntesis y punto y coma, pueden tener parámetros.

```
public interface IMetodos {// nombre de la interface

    void imprimir();

    int sumar();

    //... otros métodos
}
```

Nota: desde la versión Java 8 en adelante, se puede declarar métodos con cuerpo en una interface: default y static

Métodos

Al momento de codificar algún algoritmo, y es nuevo en el tema, posiblemente empezará a escribir y escribir líneas de código sin parar, de forma secuencial. Luego, posiblemente después de finalizar, encontrará muchas líneas, y esto puede ser complicado de leer, incluso, puede que repita y repitas cosas. Con los métodos, puede organizar el código. Un método puede llamar otro método, a los cuales se les llaman respectivamente: Método llamador y método llamado. Veamos en el siguiente ejemplo código escrito sin métodos y con métodos y el llamado:

Sin métodos:

```
    public static void main(String[] args) {
        int numero1 = 10; // se declara e inicializa la variable numero1
        int numero2 = 20; // se declara e inicializa la variable numero2
        int numero3 = 30; // se declara e inicializa la variable numero3

        int suma = numero1 + numero2;

        int resta = numero1 - numero2;

        int suma2 = numero2 + numero3;
    }
```

Con métodos:

```
public static void main(String[] args) {
        int numero1 = 10; // se declara e inicializa la variable numero1
        int numero2 = 20; // se declara e inicializa la variable numero2
        int numero3 = 30; // se declara e inicializa la variable numero3

        int suma = sumar(numero1, numero2);// llamado de método sumar
        int resta = restar(numero1, numero2); //llamado de método restar
        int suma2 = sumar(numero2, numero3);// se puede reutilizar el método sumar
        // ...
    }
```

```
public static int sumar(int num1, int num2){// los valores dentro del paréntesis se llaman parámetros
    return num1 + num2;
}

public static int restar(int num1, int num2){
    return num1 - num2;
}
```

Los métodos van a permitir, organizar su código, y evitar reescribir innecesariamente en muchas ocasiones, porque puede reutilizarlos.

En Java, como en otros lenguajes de programación, los métodos pueden retornar algún valor, usando para ello, la palabra reservada “return”, el tipo de datos que coloquemos en el return, debe ser el tipo de datos del método

Ejemplo de método con void:

```
public void imprimir(){
      System.out.println(“Hola!!!”);// imprime en consola
      return; // return es opcional y redundante para métodos void
 }
```

Variables y Constantes

Una constante, como su nombre lo dice, no cambia. Una variable es un espacio en memoria en un programa. Las variables en Java tienen varios tipos de datos primitivos, como se ilustra a continuación: