FUNDAMENTOS Y SINTAXIS DEL LENGUAJE

En la práctica anterior descargamos, instalamos y probamos el lenguaje de programación Java. Ahora vamos a ver los fundamentos y sintaxis del lenguaje.

En Java, lo primero que se construye es la clase ya que es la plantilla que declara las variables y los métodos. Al crear objetos, cada objeto será sacado de la plantilla o clase y, por lo tanto, tendrá una copia de las variables y de los métodos.

Por lo anterior concluimos que todo el código siempre va dentro de una clase (excepto lo referente a la declaración e importación de paquetes que veremos más adelante). La sintaxis de una clase es la siguiente:

```
package <package_name>;

import <other_packages>;

public class ClassName {
    <variables(also known as fields)>;

    <constructor(s)>;

    <other methods>;
}
```

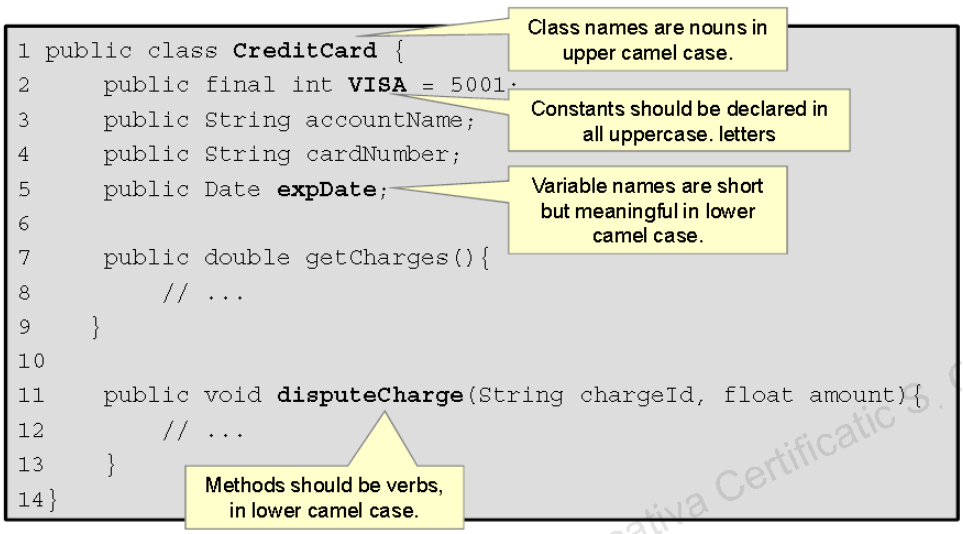
Para ejecutar una clase es necesario que tenga un método main. La cabecera del método main es la siguiente:

```
public class Simple {

    public static void main(String args[]){

    }
}
```

La convención de nombres de Java es la siguiente:



Java toma su sintaxis del lenguaje C/C++. Leer el manual parte 1 desde la página 48 hasta la página 56, que abarca los siguientes temas:

- Operators
- Logical Operators
- if else
- switch
- while
- for
- Array and for-each
- String

Este rango de páginas contiene 6 programas los cuales deben entregar para la próxima clase, de manera impresa. Por cada programa entregar código fuente e impresión de pantalla de la ejecución.

USO DEL CONSTRUCTOR DE UNA CLASE

Una clase Java representa un concepto del mundo real. Por ejemplo, regresando a la representación del concepto Alumno, modifica el archivo **Alumno.java** para que contenga lo siguiente:

```
public class Alumno
{
	// Estos atributos no tienen valor en este momento.
	// Cada objeto se encargará de darle un valor específico
	// Por lo que se llaman variables de instancia
	String numeroDeCuenta;
	String nombre;
	int semestre;

	public Alumno()
	{
	}

	public static void main(String args[])
	{
		Alumno alumno1 = new Alumno();
		alumno1.numeroDeCuenta = "111111111";
		alumno1.nombre = "Juan";
		alumno1.semestre = 3;

		Alumno alumno2 = new Alumno();
		alumno2.numeroDeCuenta = "222222222";
		alumno2.nombre = "Jorge";
		alumno2.semestre = 4;

		System.out.println("numero de cuenta:" + alumno1.numeroDeCuenta);
		System.out.println("nombre:" + alumno1.nombre);
		System.out.println("semestre:" + alumno1.semestre);
		System.out.println("----------------------");
		System.out.println("numero de cuenta:" + alumno2.numeroDeCuenta);
		System.out.println("nombre:" + alumno2.nombre);
		System.out.println("semestre:" + alumno2.semestre);
	}
}
```

Recuerda: Java siempre necesita un constructor para crear objetos para reservar espacio en memoria para cada objeto que está creando. Si el constructor no existe explícitamente, Java agrega uno implícito que, aunque no se ve, ahí está y lleva paréntesis vacíos ().

Observa que ya agregamos un constructor explícito pero que está vacío.

Compila y ejecuta esta versión. Observa el resultado.
Ahora agrega el siguiente código entre las llaves del constructor:

```
System.out.println("Creando un objeto");
```

1. Compila y ejecuta esta versión. Observa el resultado. ¿En que cambió con respecto al resultado anterior? _____

Recuerda: Además de crear el objeto, el constructor también puede inicializar las variables de instancia del objeto que se está creando. De esa manera no tenemos que inicializar cada variable en una sentencia aparte.

Modifica la clase para que ahora se observe así:

```
public class Alumno
{
	// Estos atributos no tienen valor en este momento.
	// Cada objeto se encargará de darle un valor específico
	// Por lo que se llaman variables de instancia
	String numeroDeCuenta;
	String nombre;
	int semestre;

	/*
	public Alumno()
	{
	 System.out.println("Creando un objeto");
	}
	*/

	public Alumno(String numCuenta, String nom, int sem)
	{
	 System.out.println("Creando un objeto");
	 numeroDeCuenta = numCuenta;
	 nombre = nom;
	 semestre = sem;
	}

	public static void main(String args[])
	{
		Alumno alumno1 = new Alumno();
		alumno1.numeroDeCuenta = "111111111";
		alumno1.nombre = "Juan";
		alumno1.semestre = 3;

		Alumno alumno2 = new Alumno();
		alumno2.numeroDeCuenta = "222222222";
		alumno2.nombre = "Jorge";
		alumno2.semestre = 4;

		System.out.println("numero de cuenta:" + alumno1.numeroDeCuenta);
		System.out.println("nombre:" + alumno1.nombre);
		System.out.println("semestre:" + alumno1.semestre);
		System.out.println("----------------------");
		System.out.println("numero de cuenta:" + alumno2.numeroDeCuenta);
		System.out.println("nombre:" + alumno2.nombre);
		System.out.println("semestre:" + alumno2.semestre);
	}
}
```

2. Compila esta versión. Explica por qué arroja eso el compilador

_____

Recuerda: Cuando creamos un constructor explícito con argumentos/parámetros, Java anula el constructor default.

Modifica la clase, eliminando los símbolos de comentario /* */ para que ahora se observe así y no arroje errores de compilación. Observa las 2 sentencias de creación de objetos y observa también que ya se eliminaron las sentencias de inicialización de las variables de instancia de ambos objetos:

```
public class Alumno
{
	// Estos atributos no tienen valor en este momento.
	// Cada objeto se encargará de darle un valor específico
	// Por lo que se llaman variables de instancia
	String numeroDeCuenta;
	String nombre;
	int semestre;

	public Alumno()
	{
	 System.out.println("Creando un objeto");
	}

	public Alumno(String numCuenta, String nom, int sem)
	{
	 System.out.println("Creando un objeto");
	 numeroDeCuenta = numCuenta;
	 nombre = nom;
	 semestre = sem;
	}

	public static void main(String args[])
	{
		Alumno alumno1 = new Alumno ("111111111","Juan",3);

		Alumno alumno2 = new Alumno ("222222222","Jorge",4);

		System.out.println("numero de cuenta:" + alumno1.numeroDeCuenta);
		System.out.println("nombre:" + alumno1.nombre);
		System.out.println("semestre:" + alumno1.semestre);
		System.out.println("----------------------");
		System.out.println("numero de cuenta:" + alumno2.numeroDeCuenta);
		System.out.println("nombre:" + alumno2.nombre);
		System.out.println("semestre:" + alumno2.semestre);
	}
}
```

3. ¿A cuál de los 2 constructores están invocando los 2 objetos?

_____

Agrega la creación de un tercer objeto inmediatamente después de la sentencia de creación del segundo objeto:

```
Alumno alumno3 = new Alumno();
```

Y agrega la impresión de sus variables/atributos después de la impresión de las variables/atributos del segundo objeto.

Entregar esta última versión de Alumno con la impresión de la salida de la ejecución.