**USO DE VALORES EXTERNOS**

**Qué son los valores externos**

Desde la línea de comandos se pueden pasar parámetros al programa, estos llegarán en forma de arreglo de cadena de caracteres al método main. Estos parámetros pueden ser utilizados para generar configuraciones de corrida o de instalación del programa.

En el NetBeans, se puede configurar en el proyecto configurado como principal, desde las propiedades del proyecto, running project, arguments.

**Implementación**

Este mismo método desde la línea de comandos sería:

 java -classpath aplicación.jar ar.com.educacionit.Programa 1000 2000 3000

Y el programa quedaría de la siguiente forma:

public class Programa{

  public Programa(){

  }

  public static void main(String[] args){

        for(int i = 0; i<args.length; i++){

          System.out.println("Argumento Indice:"+i+"  Valor:"+args[i]);

        }

  }

}

En el resultado de la salida de este programa, imprime:

Argumento Indice:0 Valor:1000

Argumento Indice:1 Valor:2000

Argumento Indice:2 Valor:3000

**USO DE NETBEANS**

**Vistas de un proyecto**

Las vistas son distintas formas de ver un proyecto. Project View es una vista lógica que representa al proyecto como paquetes, clases, métodos y atributos. FilesView es una vista física de los archivos y directorios que conforman al proyecto.

**Directorios de un proyecto**

El NetBeans genera la siguiente estructura de directorios para manejar el desarrollo de un proyecto:

\src, contiene el código fuente, los archivos Clase.java

\build, se genera en la compilación del proyecto y contiene el código compilado, los Clase.class

\dist, es el directorio donde se alojan los archivos para distribuir, tales como archivos miprograma.jar, o miaplicacionweb.war de forma tal que sea simple llevarlas a un ambiente productivo.

\nbproject, contiene archivos propios de la administración del proyecto de NetBeans.

**Comandos útiles aplicables a un proyecto**

Al seleccionar la vista de Project View, se puede hacer un click derecho sobre el proyecto y aparecerán los comandos más importantes, entre ellos encontraremos:

Build Project, compila el proyecto y genera el archivo aplicacion.jar

Clean Project, elimina todos los archivos compilados de la versión anterior del proyecto, dejando solo el código fuente.

Run Project, ejecuta el proyecto desde la clase principal preconfigurada, de no haber una preconfigurada mostrará las opciones de las clases con métodos main definidos.

Debug Project, se utiliza para debuguear el proyecto, hacer una corrida paso a paso viendo el contenido de las variables.

Set Main Project, configura el proyecto como el proyecto principal, entre varios proyectos.

Close Project, cierra el proyecto.

Properties, permite visualizar las propiedades del proyecto.

**El Debugger**

Herramienta que se utiliza para examinar el programa en busca de errores y depurarlos, en un paso a paso por las líneas del código fuente de toda la aplicación.

Las partes más importantes son:

Utilización de Breakpoints, se coloca un breakpoint en la línea de código donde se quiere detener la ejecución del programa para evaluar el estado de las variables.

Utilización de Watches, se utiliza para visualizar el valor de variables que uno desee y expresiones del código en ejecución.

Visualización de Local Variables, se utiliza para visualizar valores de las variables locales al código en ejecución.

**INTRODUCCIÓN A OOP**

**Qué es una clase**

**Definición**

Una clase es una agrupación de reglas de negocio, o representación de un concepto de la vida real. Es una plantilla, con reglas, para armar un objeto. Está formada por atributos, que definen un estado particular, y métodos, que definen su uso o comportamiento.

**Implementación en Java**

Ejemplos de cómo codificar diferentes clases en Java:

  public class Persona{

  }

  public class Puerta{

  }

  public class Auto{

  }

**Qué es un objeto**

**Definición**

Un objeto es una instancia de una clase. Como ejemplo, una instancia de la clase Persona, podría ser una persona llamada Mario. Puede tenerse varias instancias de una misma clase, pero una única clase de Persona, una persona Mario, y una persona María, con ambas instancias de la clase Persona.

**Implementación en Java**

Ejemplos de cómo codificar diferentes objetos en Java:

Persona mario = new Persona();

Persona maria = new Persona();

**Qué son los Atributos**

**Definición**

Los atributos determinan el estado que tiene un objeto. Estos pueden ser tipos de datos primitivos u objetos de cualquier clase.

Ejemplo:

La clase Persona tiene como atributos la altura, el color de ojos y la edad.

Ejemplo en Java:

public class Persona{

public int altura;

           public String colorDeOjos;

           public int edad;

}

**Utilización:**

Persona mario = new Persona();

mario.altura = 183;

mario.colorDeOjos = "marrones";

mario.edad = 40;

**Atributos de Instancia**

Los atributos de instancia son aplicables a un solo objeto. Determinan el estado en el que se encuentra un objeto.

Ejemplo:

El atributo altura en la clase Persona, debido a que cada persona tendrá su propia altura.

Ejemplo en Java:

  public class Persona{

        public int altura;

  }

Luego:

Persona mario = new Persona();

mario.altura = 184;

Persona maria = new Persona();

 maria.altura = 152;

**Qué son los Métodos**

**Definición**

Los métodos determinan el comportamiento y la responsabilidad que tendrán las clases. Se definen métodos que representen como se van a utilizar las clases. Este como se van a utilizar las clases significa que debemos representar el uso que se les da a las clases en la lógica de negocios de la vida real, es decir, a una puerta se le pedirá que se abra y se cierre, por lo tanto se le agregaran métodos abrir() y cerrar().

**Métodos de Instancia**

Los métodos de instancia, como su nombre lo indica, son aplicables a una instancia de la clase en particular. Es decir, que un método de instancia trabaja sobre el estado actual de la instancia, y para cada instancia tendrá un resultado distinto, por ejemplo, el hecho de comer permitirá a la instancia mario estar satisfecho, mientras que la instancia maria estará insatisfecha hasta que también se le aplique el método comer.

Ejemplo en Java:

public class Persona{

public void comer(int cantidadDeRaciones){

 // Esto es un comentario. Aqui va la definicion del metodo.

 }

}