**GUÍA DE APRENDIZAJE Nº**

1. **IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa de Formación**:  Especializaciòn tecnológica desarrollo de aplicaciones para dispositivos móviles | **Código**:  **Versión**: | 217317  1 | | |
| **Nombre del Proyecto**:  Desarrollo de Aplicaciones Móviles Para Satisfacer las Necesidades de Usuario. | **Código**: | 1019513 | | |
| **Fase del proyecto:** | | Desarrollo | | |
| **Actividad (es) del Proyecto:**  Elaborar el plan de trabajo para la construcción del sistema de información basado en los requerimientos del cliente. | **Actividad (es) de Aprendizaje:**  Conocer el lenguaje de programación y la sintaxis asociada para la futura construcción del software. | **Ambiente de formación ESCENARIO (Aula, Laboratorio, taller, unidad productiva) y elementos y condiciones de seguridad industrial, salud ocupacional y medio ambiente**  Ambiente Piso 4 # 3 del Centro de Comercio y Turismo sede cafetero; Presenta iluminación y ventilación tanto natural como artificial (ventilador). | **MATERIALES DE FORMACIÓN** | |
| **DEVOLUTIVO (Herramienta - equipo)**  Mesas: 16  Sillas **: 32**  Tablero: 1  Computadores con conectividad: 18  Televisor: 1 | **CONSUMIBLE (unidades empleadas durante el programa)**  Guía de Aprendizaje  Material Impreso y Fotocopiado |
| **Resultados de Aprendizaje**:  Realizar la codificación de los módulos del sistema y del programa principal, con el lenguaje de  programación seleccionado, de acuerdo con las especificaciones del diseño.. | **Competencia**:  Construir el sistema que cumpla con los requisitos de la solución informática. |  | | |
| **Duración de la guía ( en horas)**: | 124 |  | | |

**2. INTRODUCCIÓN**

|  |
| --- |
| Para la construcción del sistema de información se deben tener en cuenta muchos aspectos que en esta fase del proceso deben estar correctamente definidos, ellos son el análisis y diseño del sistema basado en los requerimientos del cliente.  En la fase de implementación se hace uso del documento de análisis y documento de diseño para interpretar el modelado del sistema y así por medio de un lenguaje de programación definido, iniciar con la construcción del software.  Un **Lenguaje de Programación** es aquél que es utilizado para escribir programas de computadoras que puedan ser  Entendidos por ellas. Estos lenguajes se clasifican en tres grandes categorías:  • Lenguaje de Máquina  • Lenguaje de Bajo nivel  • Lenguaje de Alto nivel  Los lenguajes de programación se utilizan para introducir en la computadora un algoritmo específico. |

1. **ESTRUCTURACION DIDACTICA DE LAS ACTIVIDADES DE APRENDIZAJE**

|  |
| --- |
| * 1. **Actividades de Reflexión inicial.**   Para iniciar con las actividades de aprendizaje de esta fase, es importante que realice una consulta exhaustiva previa, de manera autónoma sobre los conceptos de conocimiento asociados a los resultados de aprendizaje. Recuerde que tiene a su disposición 4 fuentes de conocimiento en las cuales debe indagar. (Instructor, entorno, trabajo en equipo y TIC).  Las TIC’S hacen parte importante del desarrollo mental en el ser humano a la hora de ejecutar sus labores, por esta razón y muchas más queremos fomentar esta actividad por medio de la ejecución de talleres e investigaciones a mantener su buen conocimiento sobre el tema de TIC’S. |
| * 1. **Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje.)**   Las actividades de contextualización dependen de los conocimientos previos trabajados hasta el momento, por esta razón el aprendiz deberá consultar y tener muy presente tanto los requerimientos del cliente como los artefactos resultantes de los procesos de análisis y diseño del sistema.  El aprendiz deberá exponer al grupo las necesidades del sistema de información a construir. |
| * 1. **Actividades de apropiación del conocimiento (Conceptualización y Teorización).**   **Fundamentos de Programación Java**  **Antecedentes.**  Java se creó como parte de un proyecto de investigación para el desarrollo de software avanzado para una amplia variedad de dispositivos de red y sistemas embebidos. La meta era diseñar una plataforma operativa sencilla, segura, portable, distribuida y de tiempo real.  Cuando se inició el proyecto, C++ era el lenguaje del momento. Pero a lo largo del tiempo, las dificultades encontradas con C++ crecieron hasta el punto en que se pensó que los problemas podrían resolverse mejor creando una plataforma de lenguaje completamente nueva.  Se hizo uso de la arquitectura y diseño de una amplia variedad de lenguajes como Eiffel, SmallTalk, Objetive C y Cedar/Mesa. El resultado es un lenguaje que se ha mostrado ideal para desarrollar aplicaciones de usuario final seguras, distribuidas y basadas en red en un amplio rango de entornos desde los dispositivos de red embebidos hasta su uso para soluciones en Internet.  **Características del Lenguaje**   * Lenguaje de propósito general. * Lenguaje Orientado a Objetos. * Sintaxis inspirada en la de C/C++. * Lenguaje multiplataforma: Los programas Java se ejecutan sin variación (sin recompilar) en cualquier plataforma soportada (Windows, UNIX, Mac, etc.) * Lenguaje interpretado: El intérprete a código máquina (dependiente de la plataforma) se llama Java Virtual Machine (JVM). El compilador produce un código intermedio independiente del sistema denominado *bytecode* ó código byte. * Lenguaje gratuito: Creado por SUN Microsystems, que distribuye gratuitamente el producto base, denominado JDK (Java Development Toolkit) o actualmente J2SE (Java 2 Standard Edition). * API distribuida con el J2SE muy amplia. Código fuente de la API disponible.   **Facilidades del J2SE (Java 2 Estándar Edition)**   * Herramientas para generar programas Java. Compilador, depurador, herramienta para documentación, etc. * La JVM, necesaria para ejecutar programas Java. * La API de Java (jerarquía de clases). * Código fuente de la API (Opcional). * Documentación.   **Instalación JAVA**  Java puede ser instalado basado en dos enfoques, el enfoque para desarrolladores JDK, o el enfoque para clientes JRE.  **JDK - JRE**  El **JDK** (Java Development Kit) es el software requerido para la creación de programas Java, consiste en un conjunto de librerías y herramientas de desarrollo necesarias para los procesos de compilación y ejecución de las aplicaciones.  El **JRE** (Java Runtime Environment) es el entorno mínimo para la ejecución de programas, sin esta las aplicaciones Java no podrían ser ejecutadas satisfactoriamente.  En nuestro caso nos vamos a centrar en el enfoque de desarrollador, por lo tanto instalaremos el JDK.  **Instalación JDK.**  Hay que tener en cuenta que al momento de instalar el **JDK** el **JRE** es instalado por defecto, ya que las aplicaciones se prueban primero por el desarrollador y para esto la deben ser ejecutadas, sin embargo el JRE puede ser instalado sin que el **JDK** lo esté, pues por lo regular esta herramienta se instala en máquinas donde solo es necesario ejecutar las aplicaciones mas no realizar actividades de desarrollo o compilación (instalación en la máquina del Cliente).  Para efectos del curso vamos a realizar el proceso de descarga e instalación del **JDK**, el **JRE** no nos interesará por el momento pues como se mencionó viene implícito en la instalación.  **Descarga JDK.**  Inicialmente descargamos el instalador desde la página de **Oracle** (<http://www.oracle.com/technetwork/es/java/javase/downloads/index.html>)escogemos la distribución que deseamos instalar, para este caso vamos a descargar el jdk 7.    Al ingresar al link, se carga otra página donde se deberán aceptar los términos de licencia y seleccionar el link correspondiente al sistema operativo sobre el cual se realizará la instalación.    Después de seleccionar el sistema operativo o la distribución correspondiente, se mostrará una ventana de descarga, desde allí se obtiene el **JDK** y se almacena en cualquier directorio de la máquina.    Después de descargarlo, al hacer doble clic se ejecutará el proceso de instalación.  F:\HENAO\codejavu\instalacion JDK\4. ventanas.png  Al final se cargará una última ventana informando que el proceso se ha realizado exitosamente.  **Creación Variable de entorno.**  Las variables de entorno permiten al sistema operativo conocer las rutas de instalación de ciertos programas o herramientas, y aunque inicialmente no es necesario, es una buena práctica crear la variable JAVA\_HOME para facilitar la ubicación de Java en la máquina.  Una de las maneras de crear la variable de entorno es presionando inició, clic derecho en equipo, posteriormente se da clic en propiedades.    Al hacer esto se cargan las propiedades de la máquina, ahora se da clic en Configuración avanzada del sistema.    Se carga una ventana donde presionamos el botón “Variables de entorno…” y acto seguido en la ventana resultante se presiona el botón “Nueva” y se introduce el nombre de la variable y la ruta de instalación del JDK.    Y de esta manera ya finalizamos nuestro proceso de instalación de Java en nuestras maquinas.  **Entorno de Desarrollo.**  Un entorno de desarrollo es el ambiente necesario para desarrollar aplicaciones, si vamos a trabajar con Java existen diferentes entornos que nos permiten hacerlo, estos son conocidos como IDE (integrated development environment), para el desarrollo del curso vamos a trabajar con el IDE ECLIPSE en cualquiera de sus versiones.  Para la descarga e instalación de Eclipse nos podemos guiar con el siguiente enlace: <http://codejavu.blogspot.com/2013/04/configuracion-ambiente-de-desarrollo.html>  **Entorno Básico.**  Después de haber instalado java y teniendo el IDE en nuestra máquina, podemos abrirlo y visualizar lo siguiente.    La ventana anterior corresponde a la selección del workspace o entorno de trabajo donde se alojarán nuestros proyectos java, corresponde básicamente a la ruta de la carpeta donde queremos guardar los proyectos o aplicaciones.  Al definir la ruta, se carga la ventana inicial la primera vez con esta forma.    Esta es la página de bienvenida del IDE, dependiendo de la versión de Eclipse, esto puede variar, aquí simplemente podemos cerrar la página desde la x en la esquina superior izquierda al lado de **Welcome**.  Al hacerlo se carga la siguiente estructura, correspondiente a los paneles principales de nuestro entorno de desarrollo.    **Panel Explorador de Proyectos:** Este panel permite visualizar los proyectos con los que estamos trabajando, inicialmente arranca en blanco pues no tenemos creado ningún proyecto, pero a medida que avancemos aquí se cargarán.  **Panel para el trabajo con clases:** este panel será el espacio de trabajo donde visualizaremos las clases con las que vamos a trabajar.  **Panel Elementos:** este panel permite visualizar la estructura interna de nuestras clases, se despliegan los métodos, variables, tipos de datos entre otra información en general.  **Consola:** en este panel se verán los procesos ejecutados por nuestro sistema, si el desarrollador desea imprimir algún valor, se cargará aquí, también permite evidenciar errores, advertencias o mensajes de compilación.  **Nuestro Primer Proyecto en Java.**  Java es un lenguaje de programación Orientado a Objetos, para desarrollar programas en este lenguaje es importante tener claro los conceptos básicos de la POO, sin embargo por el momento vamos a conocer lo básico del trabajo con Java, más adelante nos adentraremos en el mundo de la Programación Orientada a Objetos.  Para iniciar debemos saber que java se compone de clases, estas clases son ficheros tomados como plantillas para determinar los procesos lógicos o algoritmos que componen el programa, la estructura básica de una clase es la siguiente.  Public Class NombreClase  {  //Conjunto de instrucciones:  }  Vamos a iniciar con nuestro primer programa Java desde Eclipse, creando un proyecto con una clase principal.  Para iniciar vamos a **File/new/Java Project**    Al hacerlo se carga una ventana donde definimos el nombre del proyecto.    Al darle finish, se crea una carpeta del proyecto con el nombre introducido.    Con el mouse nos paramos en el proyecto, se da clic derecho/New/Class    Se carga la siguiente ventana, donde en la sección de nombre se introduce el nombre de la Clase.    Al darle Finish, se crea un archivo .java con el nombre de la clase y la estructura anterior.    Dentro de la clase Principal escribimos lo siguiente.  **public** **static** **void** main (String arg[])  {  System.*out*.println("Hola Aprendices SENA");  }  Quedando algo así    Para ejecutar la aplicación se le debe dar clic al icono “Run” en la barra de herramientas    Al hacer esto en la consola se mostrará el mensaje de Bienvenida.    Y de esta manera ya tenemos nuestra primera aplicación java creada.  Nota: como se pudo observar se utilizó la línea **System.out.println();** la cual permite imprimir mensajes en consola, la información que se quiera imprimir debe estar dentro de los paréntesis, si es texto se escribe entre comillas, si se desean imprimir datos almacenados en variables, tan solo se escribe el nombre de la variable.  **Proceso de Compilación y Ejecución de un programa Java.**  Al realizar el ejercicio anterior, podemos ver que la construcción de un programa Java tiene diferentes etapas:  **Primera Etapa.**   * Digitar el programa: Guardando el archivo con extensión .java (Principal.java)   **Segunda etapa.**   * Compilar el programa. * Corregir errores de sintaxis. * Se genera el código intermedio o byteCode en los archivos punto class (Principal.class).   **Tercera etapa.**   * En la etapa de ejecución del programa, la Máquina Virtual de java (JVM), interpreta las instrucciones byteCode. * Posteriormente se realiza otra serie de pasos internos en el computador, se presenta el resultado del programa ya sea en pantalla o dependiendo del proceso que se desea realizar.   Estas etapas se pueden evidenciar de la siguiente manera.      **Conceptos Básicos.**  Para el trabajo con Java, debemos conocer algunos conceptos básicos a tener en cuenta, entre ellas las convenciones de código.  Las convenciones permiten definir una serie de estándares para el trabajo con el lengueja de programación, estas definen la forma como se debe programar, facilitando mucho más el entendimiento del código fuente y los lineamientos para la creación de aplicaciones. En el material de estudio de la plataforma se puede encontrar la guía de convenciones para el lenguaje de programación, también se puede consultar las siguientes direcciones:  **Que son las convenciones:** <http://codejavu.blogspot.com/2014/04/que-son-las-convenciones-de-codigo.html>  **Ej. Convenciones de código:** <http://codejavu.blogspot.com/2014/04/convenciones-de-codigo-en-java.html>  **Palabras Reservadas.**  Java define una serie de palabras para la identificación de operaciones, métodos, clases y demás elementos, con el fin de que el compilador pueda entender los procesos que se están desarrollando.    Las palabras reservadas no pueden ser usadas por el desarrollador para nombres de métodos, variables, clases entre otras, pues como se mencionó cada una tiene un objetivo dentro del lenguaje.  **Variables en Java.**  Una variable es un contenedor que puede almacenar información y puede cambiar en el tiempo, pues su contenido puede variar, básicamente se puede definir como un nombre que identifica una dirección de memoria.  Las variables en java se componen de un identificador y un tipo de dato que lo acompaña.  **<TipoDato> identificador;**  **Identificadores**: Representan la manera correcta de definir nombres de variables, para crear los identificadores se debe ajustar a la convención de código.   * El primer carácter debe ser un carácter alfabético (a…z, A…Z) o $, \_ * Después del primer carácter pueden ir caracteres alfanuméricos (a…z, A…Z) , (0…9) o $, \_ * Los identificadores no pueden ser palabras reservadas. * Por convenciones, se recomienda que aplique la regla camelCase   **Tipos de Datos.**  En java existen 2 tipos de datos, estos se conocen como tipos de datos primitivos y de referencia.  **Tipos de datos primitivos**: se caracterizan por tener un único valor, estos son representados en la siguiente tabla.    Los tipos de datos tienen un tamaño definido por un rango basado en el número de bytes que le corresponde, estos bytes definen el espacio en memoria que ocupan al ser utilizados.  **Datos tipo referencia**: Son variables correspondientes a los arreglos, clases e interfaces. Se crean con la instrucción new y serán estudiados durante el desarrollo del curso.  Como se mencionó, las variables son valores que se pueden cambiar y corresponden a nombres que representan un valor de cierto tipo (int, char, float, doublé, entre otros). El valor asociado al nombre puede variar. Las variables corresponden a localidades de memoria que se dónde se almacenarán los valores dados.  **Declaración de Variables.**  Para poder usar una variable, esta debe ser previamente declarada, así ajustándonos a la estructura anterior (tipoDato identificador) podemos crear nuestra variable de la forma: tipoDato nombreVariable así.  int x; //corresponde a una variable de tipo int (entera) con el nombre x.  double count; //se declara una variable double llamada count  char a,b,c; //se declaran 3 variables de tipo char llamadas a, b y c  **Inicialización de Variables.**  Después de haber declarado las variables, podemos inicializarlas, para eso usamos el operador =, lo cual permite definir cuál es el valor que va a tomar o almacenar la variable.  Posibles valores que puede almacenar una variable dependiendo de su tipo.    Para las variables de tipo de dato char, como se mencionó, trabajan con datos Unicode de 16 bits, para saber los valores equivalentes se usa la tabla de códigos ascii  D:\videoTutoriales\2\codigo_ascii.gif  **Variables Finales (Constantes)**  Las constantes corresponden a datos que una vez inicializadas no pueden cambiar su valor, en java estas son definidas usando la palabra reservada “final” antes del tipo de dato y posteriormente el nombre de la constante (por convenciones este nombre debe ser en mayúscula)  **final <tipoDato> IDENTIFICADOR;**  **Ej:** final double PI = 3.141516;  final INT VALOR;  Se debe tener en cuenta que las reglas para convenciones de variables aplican igual para las convenciones de constantes excepto por como ya se mencionó el nombre de la constante que va en mayúsculas y si hay más de una palabra, estas serán separadas por guion bajo (final doublé PORCENTAJE\_DESCUENTO).  **Operadores Aritméticos.**  Estos operadores corresponden a los usados para labores académicas cotidianas tales como procesos de sumas, restas, división, multiplicación, modulo, incremento, decremento, su aplicación se puede evidenciar en la siguiente tabla.   |  |  |  | | --- | --- | --- | | **Operador** | **Uso** | **Descripción** | | + | Operador1 + operador2 | **Suma** (Operador1 mas Operador2) | | - | Operador1 - operador2 | **Resta** (Operador1 menos Operador2) | | \* | Operador1 \* operador2 | **Multiplicación** (Operador1 por Operador2) | | / | Operador1 / operador2 | **División** (Operador1 dividido Operador2) | | % | Operador1 % operador2 | **Módulo-residuo** (Operador1 Modulo Operador2) | | ++ (postfijo) | Operador++ | **Incremento Postfijo**, incrementa en 1 el operador, pero primero evalúa antes de realizar el incremento. (obtiene el valor e incrementa) | | ++  (prefijo) | ++Operador | **Incremento Prefijo**, incrementa en 1 el operador, primero incrementa y luego evalúa. (incrementa y luego obtiene el valor) | | --  (postfijo) | Operador-- | **Decremento Postfijo**, decrementa en 1 el operador, pero primero evalúa antes de realizar el decremento. (obtiene el valor y decrementa) | | --  (prefijo) | --Operador | **Decremento Prefijo**, decrementa en 1 el operador, primero decrementa y luego evalúa. (decrementa y luego obtiene el valor) |   Ej: realizar las siguientes operaciones para encontrar el valor de Z, cuando X=10 y Y=2;   * **Z=X+Y;**  suma X **más** Y almacenando 12 en Z * **Z=X-Y;** resta X **menos** Y almacenando 8 en Z * **Z=X\*Y;** multiplica X **por** Y almacenando 20 en Z * **Z=X/Y;** divide X **dividido** Y almacenando 5 en Z * **Z=X%Y;** obtiene el residuo de la operación X/Y, se lee X **modulo** Y almacenando 0 en Z * **Z=X++;** incrementa X, primero obtiene 10 y luego lo incrementa en 11, almacenando 11 en Z * **Z=++X;** incrementa X, primero incrementa a 11 directamente y luego obtiene 11, almacena 11 en Z * **Z=X--;** decrementa X, primero obtiene 10 y luego lo decrementa en 9, almacenando 9 en Z * **Z=--X;** decrementa X, primero decrementa a 9 directamente y luego obtiene 9, almacena 9 en Z   **NOTA:** Tenga en cuenta que cuando se usan los incrementos o decrementos postfijos o prefijos, la lógica cambia, es decir, si se quiere imprimir x++, el resultado será 10 en la línea de impresión, pero si se imprime ++x, el resultado será 11 directamente.  **Ejercicio.**  Cree un proyecto en Eclipse y en él una clase con el siguiente código:  **NOTA:** analice el cambio al utilizar los operadores postfijos y Prefijo, imprima 2 mensajes después de realizar los incrementos y decrementos, verifique su funcionamiento.  **Precedencia de operadores.**  La precedencia de operadores indica la forma correcta de resolver una operación matemática en caso de que no se use el operador de agrupación (paréntesis “()” ), para eso se utilizan los operadores aritméticos vistos anteriormente.    Se resuelven primero los operadores de mayor precedencia, en caso de tener igual precedencia se resuelven de izquierda a derecha  Ej: Resolver 10/5+6-2+3\*8\*1-12/2+9-7\*4    **Operadores lógicos y relacionales.**  Las condiciones son definidas mediante expresiones lógicas o relacionales, dichas expresiones utilizan operadores que permiten obtener un resultado en función de verdadero o falso.  **Operadores relacionales.**  Los operadores relacionales permiten definir la relación entre 2 o más expresiones.   |  |  | | --- | --- | | **Operador** | **Descripción** | | < | Menor Que | | > | Mayor Que | | == | Igual Que | | <= | Menor o Igual Que | | >= | Mayor o Igual Que | | != | Diferente De |   La forma de usar e interpretar estos operadores es usando la estructura:  Expresión1 **OperadorRelacional** Expresión2  Donde el resultado de la operación anterior se evalúa en términos de verdadero o falso.  **Ejemplo: evalúe las siguientes condiciones para A=4**   |  |  |  | | --- | --- | --- | | **Condición** | **Resultado** | **Descripción** | | A < 4 | FALSO | A vale 4 por lo tanto no es Menor Que 4 | | A > 4 | FALSO | A vale 4 por lo tanto no es Mayor Que 4 | | A == 4 | VERDADERO | A vale 4 por lo tanto si es Igual Que 4 | | A <= 4 | VERDADERO | A vale 4 por lo tanto aunque no es menor que 4, si es Igual | | A >= 4 | VERDADERO | A vale 4 por lo tanto aunque no es Mayor que 4 si es Igual | | A != 4 | FALSO | A vale 4 por lo no es Diferente De 4 |   **Ejemplo: evalúe las siguientes condiciones para A=5**   |  |  |  | | --- | --- | --- | | **Condición** | **Resultado** | **Descripción** | | A < 4 | FALSO | A vale 5 por lo tanto no es Menor Que 4 | | A > 4 | VERDADERO | A vale 5 por lo tanto si es Mayor Que 4 | | A == 4 | FALSO | A vale 5 por lo tanto no es Igual Que 4 | | A <= 4 | FALSO | A vale 5 por lo tanto no es menor que ni es Igual a 4 | | A >= 4 | VERDADERO | A vale 5 por lo tanto si es Mayor que 4 aunque no es Igual | | A != 4 | VERDADERO | A vale 5 por lo sí es Diferente De 4 |   **Ejercicio.**  Cree un proyecto en Eclipse y en él una clase con el siguiente código:    Después de verificar la salida del programa, haga cambios asignándole un nuevo valor a x o a y, luego pruebe nuevamente, realice el cambio mediante 3 pruebas y explíquelo.  **Operadores lógicos.**  Los operadores lógicos permiten la combinación de condiciones para formar una sola expresión lógica, para estos permiten obtener un resultado lógico al complementarse con los operadores relacionales.  La forma de usar e interpretar estos operadores es usando la estructura :  operando1 **OperadorLógico** operando2   |  |  |  | | --- | --- | --- | | **Operador** | **Nombre** | **Descripción** | | ! | Negación (NO) | Al usar este operador si la condición arroja falso, el operador la convierte a verdadero y viceversa | | && | Conjunción (Y) | Es verdadero solo si los 2 (o todos) los operando son verdadero, caso contrario arroja falso | | || | Disyunción (O) | Es verdadero cuando al menos un operando es verdadero |   Ejemplo: evalúe el resultado de los siguientes operaciones, para los operando A y B, asuma que cada operando es el posible resultado de una operación relacional.  **A y B son valores Booleanos**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **A** | **B** | **!A** | **!B** | **A && B** | **A || B** | | F | F | V | V | F | F | | V | F | F | V | F | V | | F | V | V | F | F | V | | V | V | F | F | V | V |   **Ejercicio.**  Cree un proyecto en Eclipse y en él una clase con el código de la imagen, verifique el uso de cada operador, haga pruebas con 5 valores o condiciones diferentes, verifique el funcionamiento y explique lo sucedido. |
| * 1. **Actividades de transferencia del conocimiento.**  1. Evalúe cuál de las siguientes expresiones es verdadera y cual es falsa (true o false)   a. 8 <= 5  b. 3 > 6  c. 4 != 7  d. b == b + 1  e. c++ != c++  f. 19 == 19   1. Determine por que el resultado del siguiente enunciado es falso (false).   (7 > 10 || 10 >2 && 3 == 4)     1. Qué valor lógico queda almacenado en x.   int y=5;  boolean x=(y>4)&&(y<7)   1. Realice un programa que permita calcular el sueldo final de un empleado, tenga en cuenta que el sueldo depende de un incremento del 30%, haga el algoritmo definiendo el porcentaje como una constante. 2. Haga un algoritmo que permita imprimir el resultado del ejemplo propuesto sobre la precedencia de operadores. 3. Modifique el algoritmo anterior, introduciendo cada valor en una variable independiente. |
| * 1. **Actividades de evaluación.**  |  |  |  | | --- | --- | --- | | **Evidencias de Aprendizaje** | **Criterios de Evaluación** | **Técnicas e Instrumentos de Evaluación** | | **Evidencias de Producto:** | Conoce los fundamentos básicos de programación | Lista de chequeo sobre conocimientos básicos, teóricos y prácticos. | |

1. **RECURSOS PARA EL APRENDIZAJE**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ACTIVIDADES DEL PROYECTO** | **DURACIÓN  (Horas)** | **Materiales de formación devolutivos: (Equipos/Herramientas)** | | **Materiales de formación (consumibles)** | | **Talento Humano (Instructores)** | | **AMBIENTES DE  APRENDIZAJE TIPIFICADOS** |
| **Descripción** | **Cantidad** | **Descripción** | **Cantidad** | **Especialidad** | **Cantidad** | **ESCENARIO (Aula, Laboratorio, taller, unidad productiva)** y elementos y condiciones de seguridad industrial, salud ocupacional y medio ambiente |
| Planeación | 124 | Mesas  Sillas  Tablero  Equipos de cómputo con mínimo 2 GB de RAM, paquetes ofimáticos y conexión a internet | 23  30  1  18 | Guía de Aprendizaje | 1 | Ingeniero de Sistemas o Tecnólogo en Análisis y desarrollo de sistemas de Información, con experiencia de mínimo 24 meses en áreas de desarrollo de software | 1 | Ambiente de Formación limpio y provisto de caneca de basuras, con buena iluminación natural o artificial, con conexiones electicas en buen estado y vías de acceso o de salida de emergencia señalizadas |

|  |
| --- |
| **Sistema de Cómputo:** se utiliza para señalar lo que el usuario emplea, en lugar del término Computadora  **Sistema operativo:** ambiente en el cual los usuarios pueden preparar programas y ejecutarlos sin tener que entrar en los detalles del hardware del sistema.  **Programa:** secuencia de instrucciones que orienta a la Unidad de Central de Procesamiento (CPU) en el desarrollo de los cálculos, el cual debe expresarse de forma que sea entendido por el CPU.  **Algoritmo**: se define como un método que se realiza paso a paso para solucionar un problema que termina en un número finito de pasos  **Lenguaje de programación:**  es aquél que es utilizado para escribir programas de computadoras que puedan ser  **Lenguaje de Maquina:** Son aquéllos que están escritos en lenguajes directamente inteligibles por la máquina (computadora), ya que sus instrucciones son cadenas binarias (cadenas o series de caracteres de dígitos 0 y 1) que especifican una operación y las posiciones (dirección) de memoria implicadas en la operación se denominan instrucciones de máquina o código máquina.  **Lenguaje de Bajo Nivel:** El lenguaje de bajo nivel por excelencia es el ensamblador. Las instrucciones en lenguaje ensamblador son instrucciones conocidas como nemotécnicos.  **Lenguaje de Alto Nivel:** Están diseñados para que las personas escriban y entiendan los programas de un modo mucho más fácil que los lenguajes máquina y ensambladores. |

1. **GLOSARIO DE TERMINOS**
2. **REFERENTES BIBLIOGRÁFICOS**

|  |
| --- |
| Fundamentos de programación. Problemas resueltos de programación en lenguaje Java. José María Pérez Menor, Jesús Carretero Pérez, Félix García Carballeira, and José Manuel Pérez Lobato. Madrid: Paraninfo, 2003. p[1]-22  Programación orientada a objetos. Problemas resueltos de programación en lenguaje Java. José María Pérez Menor, Jesús Carretero Pérez, Félix García Carballeira, and José Manuel Pérez Lobato. Madrid: Paraninfo, 2003. p[79]-114  Arrays y agrupaciones de objetos. Problemas resueltos de programación en lenguaje Java. José María Pérez Menor, Jesús Carretero Pérez, Félix García Carballeira, and José Manuel Pérez Lobato. Madrid: Paraninfo, 2003. p[115]-147.  [**http://www.alegsa.com.ar/Dic/gui.php#sthash.rhWG25ly.dpuf**](http://www.alegsa.com.ar/Dic/gui.php#sthash.rhWG25ly.dpuf)  [**http://www.sedic.es/autoformacion/accesibilidad/2-definicion-accesibilidad.html**](http://www.sedic.es/autoformacion/accesibilidad/2-definicion-accesibilidad.html)  [**http://www.sidar.org/recur/desdi/traduc/es/visitable/quees/usab.htm**](http://www.sidar.org/recur/desdi/traduc/es/visitable/quees/usab.htm)  [**http://usabilidadaldia.com/usabilidad**](http://usabilidadaldia.com/usabilidad)  [**http://www.hipertexto.info/documentos/interfaz.htm**](http://www.hipertexto.info/documentos/interfaz.htm) |

1. **CONTROL DEL DOCUMENTO (ELABORADA POR)**

|  |
| --- |
| **Cristian David Henao H.**  Ingeniero de Sistemas y Computación.  Centro de Comercio y Turismo – Sena Regional Quindío  Instructor Contratista  Febrero de 2016 |