Tests de Autoevaluación: Introducción a Java y Elementos Básicos del Lenguaje

Alumno: Rodrigo Sosa

Fecha: 21 / 9 / 2024

Test de autoevaluación: Introducción a Java

1) ¿Qué es el código de ensamblaje y el programa traductor ensamblador?

El código de ensamblaje era una notación simbólica que representaba las operaciones que podía realizar la computadora. El programa traductor ensamblador es un programa que traduce el código de ensamblaje a código de máquina.

2) ¿Qué es la abstracción en informática?

La abstracción en informática se crea cuando el programa traductor ensamblador ignora selectivamente ciertas partes de un todo en el código para un mejor entendimiento.

3) Explique los diferentes niveles de abstracción en lenguajes de programación y de algunos ejemplos de lenguajes por nivel de abstracción.

Nivel Bajo: Este nivel mantiene un bajo con el objetivo de que el programador escriba líneas de código en un lenguaje más cercano al humano, y tenga una flexibilidad mayor a la hora de programar, ocupándose más del problema en sí a resolver que en el hardware sobre el cual tiene que programar. Ej: Lua.

Nivel Medio: Los lenguajes de programación de medio nivel de abstracción como BCPL, C y Basic, están en un intermedio entre los lenguajes de bajo nivel y los de alto nivel. Si bien estos lenguajes a menudo se los considera como de alto nivel, poseen la característica de poder programar directamente sobre el hardware como se hacía en lenguajes de bajo nivel y al mismo tiempo poder realizar operaciones de alto nivel.

Nivel Alto: Los lenguajes de alto nivel están orientados a objetos, están orientados a procedimientos, los cuales se componen de procesos. Ej: PHP, Java y C + +.

4) Defina intérprete, compilador y explique la diferencia entre estos dos conceptos.

Un intérprete es un programa que realiza la traducción a medida que sea necesaria, línea por línea, y por lo general no guardan el resultado de la traducción. Permite que el código se ejecute en cualquier sistema, aunque la desventaja es que el intérprete se encontrará cargado en memoria todo el tiempo para poder hacer esta traducción, factor que hace que un programa interpretado sea más lento que uno compilado.
Un compilador es un programa que, a diferencia del intérprete, realiza toda la traducción de un archivo fuente de una sola vez. La ventaja es que al no tener cargado un intérprete en la memoria, el programa será más rápido, pero la desventaja es que el ejecutable generado por el compilador sólo funcionará en el sistema específico para el que se haya compilado el código.

5) ¿Qué es un paradigma de programación? Nombre algunos de ellos.

Los paradigmas de programación son maneras o estilos de programación empleados, son enfoques propuestos para resolver problemas, los cuales tienen una aceptación por una comunidad de programadores. Algunos de ellos son: el imperativo, el lógico, el declarativo, el orientado a eventos, y el orientado a objetos.

6) Explique que es el paradigma orientado a objetos

El paradigma orientado a objetos se basa en el manejo de datos mediante una entidad llamada objetos la cual a su vez puede relacionarse con otras entidades de este tipo, es el paradigma más usado por la gran mayoría de los programadores dadas sus ventajas ya que este paradigma es muy flexible, presenta una manera de resolver problemas muy grandes en forma ordenada y sencilla, y presenta características para la reutilización de código.

7) ¿Cómo es que Java es tan popular en tecnologías webs, si cuando se creó el lenguaje no existía Internet tal y como se la conoce en la actualidad?

Es tan popular porque fué diseñado para crear una interfaz atractiva e intuitiva para electrónica de consumo y que en ese momento era llamado Green, aunque eso no funcionó. Luego de ver que el lanzamiento del reciente navegador web gráfico Mosaic, el equipo apostó a que internet evolucionaría como habían pensado que lo haría la tecnología de consumo, y fue así que en 1994 reorientaron el lenguaje a plataformas web y renombraron finalmente el lenguaje, llamándose ahora JAVA.

8) Liste las características más relevantes del lenguaje y desarrolle brevemente cada una de ellas.

Las características son:

Paradigma orientado a objetos: Es el paradigma por excelencia, adoptado por la mayoría de programadores del mundo en la actualidad. Trata sobre el manejo de datos conjunto a sus operaciones en una entidad llamada objeto, la cual puede relacionarse con otros objetos. El mismo proporciona independencia de código y por lo tanto la posibilidad de reutilizar el mismo, lo cual permite construir grandes proyectos de una manera sencilla y ordenada.

Facilidad de uso: Java combina ser un poderoso lenguaje y tener una gran facilidad de uso ya que el mismo está basado en lenguajes como C y C++. Como estos lenguajes son muy populares, y por tanto conocidos por la mayoría de programadores, hacen que al sumergirse en JAVA el código resulte familiar, sumado a las facilidades de uso que proporciona el propio lenguaje.

Independencia de plataforma: Java es un lenguaje que está diseñado para que sus programas se ejecuten bajo una máquina virtual. Gracias a esto es posible que un mismo código compilado pueda ser ejecutado en múltiples plataformas sin tener que recompilar el código para cada plataforma en la cual se quiera ejecutar un programa.

Soporte para trabajo en red: Java ofrece una total transparencia a la hora de trabajar en red. Para eso se abstrajeron las características y detalles específicos en referencia a los aspectos técnicos de redes de comunicaciones. Con el objetivo de simplificar la tarea del programador a la hora de trabajar con redes a bajo nivel, Java proporciona formas básicas de trabajar en red mediante Sockets y URL.

Seguridad en la ejecución de aplicaciones en sistemas remotos: Al ser Java un lenguaje multiplataforma, ofrece la posibilidad de programar aplicaciones que pueden ser ejecutadas en navegadores web. Una aplicación programada para ejecutarse en un navegador web podrá por lo tanto ejecutarse en sistemas remotos. A diferencia de Microsoft, Java presenta un sistema de seguridad por capas, el cual se basa en certificados y restringe acciones como el acceso directo al disco rígido y al portapapeles.

9) ¿Cómo es que Java es multiplataforma y al mismo tiempo sus programas se ejecutan de una manera eficiente respecto al tiempo de ejecución?

Es eficiente porque Java tiene una máquina virtual, un microprocesador virtual que trabaja con un código binario especial.

10) ¿Qué es y cómo funciona la máquina virtual de Java?

La máquina virtual trabaja con un código binario especial, llamado bytecode, el cual es generado por un compilador del lenguaje. El código resultante compilado por el lenguaje es ejecutado por la máquina virtual de JAVA, la cual por un proceso de interpretación o bien mediante un compilador JIT (Just in Time), convierte el bytecode en código nativo de la plataforma en la que se vaya a ejecutar, lo que hace a la ejecución mucho más rápida, y que permite entonces que un mismo programa pueda ser ejecutado en diferentes plataformas como Windows, Linux, Mac, etc.

Test de autoevaluación: Elementos Básicos del Lenguaje

1) ¿Qué es un linker o cargador, y para qué sirve?

Un linker o cargador es una interfaz que se encarga de suministrar un bloque de memoria sobre lo que se desee trabajar.

2) ¿Cuáles son las tres zonas de memoria utilizadas por Java?

Java trabaja con tres zonas de memoria: la zona de datos, la pila (stack) y el montículo (heap).

3) Explique para qué sirve la zona de memoria de datos.

En la zona de memoria de datos se guardan las instrucciones del programa ejecutable en código de máquina, las constantes, el código correspondiente a los métodos, y a las clases. El tamaño de ésta zona en memoria se establece en tiempo de compilación, por lo cual al compilador le es necesario saber de antemano el tamaño de cada elemento que vaya a guardar, asegurándose de que éstos no cambien en tiempo de ejecución, ya que esta zona de memoria es fija, y no variará a lo largo de la ejecución del programa.

4) Explique para qué sirve la zona de memoria conocida como pila o stack.

La zona de memoria conocida como pila o stack es una memoria más flexible y eficiente. La pila funciona de manera en que lo primero que se guarda en ella será lo último en ser eliminado, y lo último que se guarda será lo primero en ser eliminado (last in first out).La pila cuenta con una cantidad limitada de memoria, y en ella se guardan variables locales, variables de referencia, parámetros y valores de retorno, resultados parciales, y el control de la invocación y retorno de métodos.

5) Explique para qué sirve la zona de memoria conocida como montículo o heap.

La zona de memoria conocida como montículo o heap es un espacio de memoria dinámica en donde se guardarán variables de instancia y objetos. El encargado de administrar este espacio de memoria es el Garbage Collector o recolector de basura, que libera al programador de estas tareas.

6) ¿Qué es un atributo y de qué tipos de datos puede ser definido?

Los atributos son espacios de memoria donde se pueden almacenar datos. Los mismos están conformados por un nombre que sirve para identificarlos dentro del programa y pueden estar asociados a un valor. Así mismo este valor está asociado a un tipo de dato. Los atributos pueden ser de longitud fija o variable. En el caso de atributos de longitud fija, se definen indicando la cantidad de datos que pueden almacenar, y ésta cantidad no variará a lo largo de la ejecución del programa. Ej: arrays. De la misma manera existen atributos de longitud variable, en donde la cantidad de datos que pueden almacenar puede variar en la ejecución del programa. Un ejemplo de estos tipos de atributos son las colecciones de datos.

7) ¿Cuál es la diferencia entre tipos de datos primitivos y de referencia? Explique cada una de estas categorías.

Los tipos de datos primitivos son tipos de datos sencillos y pequeños, con un tamaño conocido, los cuales son colocados en la pila, para que el programa resulte más eficiente. Mientras que los tipos de datos de referencia están ligados al concepto de objetos, ya que justamente un tipo de “referencia”, hace referencia a un objeto. Por lo tanto estos tipos serán referencias a objetos, a interfaces y a arreglos. Al referenciar a estos elementos, lo que se tiene en atributos de estos tipos, es en realidad, una posición de memoria que apunta a dichos elementos, y por lo tanto, si se crean dos atributos del mismo tipo, con algún tipo de contenido, y se asigna el primer atributo al segundo atributo, no se copiará un valor, sino que se pasará una dirección de memoria, por lo que, si el segundo atributo es modificado, evidentemente esos cambios se verán reflejados también en el primero.
Tipos de datos primitivos: boolean, char, byte, short, int, long, float, double y void.
Tipos de datos de referencia: objetos, arrays e interfaces.

8) El tipo de dato String, ¿Es un tipo de dato primitivo o de referencia? Explique las particularidades correspondientes al mismo.

El tipo de datos String no es un tipo de dato primitivo, pero debido al trato especial que Java da sobre él se tiene que pensar que sí lo es, esto debido a que al encerrar una cadena de caracteres entre comillas dobles, lo que Java hace realmente es crear un objeto de tipo String. Pero tampoco se puede decir que funciona exactamente como un tipo de referencia, ya que, en realidad, cuando se declara un tipo String se la declara como constante, lo que quiere decir, que una vez indicado el valor de la cadena, no podrá modificarse.

9) ¿Qué es la conversión entre tipos de datos y en qué casos resultaría útil hacer este tipo de conversión? Detalle los problemas que pueden surgir al trabajar con conversiones.

La conversión de tipos de datos, como su nombre lo indica, sirve para convertir tipos de datos primitivos entre sí, ya sea pasar de un int a un float, o un byte a un char. Esto sirve cuando necesitamos cualidades de otros tipos de datos, como buscar un número decimal en un cálculo entre un int o float.

10) Liste y explique las convenciones de escritura adoptado por el común denominador de programadores

Clases: Las clases deben ser escritas en letras minúsculas, con la primera letra de cada palabra en mayúsculas, esta forma de escritura es llamada lomo de camello o CamelCase en inglés.

Métodos: Los métodos deben ser verbos escritos en minúsculas con la primera letra de cada palabra en mayúsculas con excepción de la primera palabra del método.

Atributos: Al igual que los métodos deben ser escritos en minúsculas con la primera letra de cada palabra en mayúsculas con excepción de la primera palabra del identificador del atributo.

Constantes: Las constantes deben escribirse en su totalidad con mayúsculas y para separar palabras se debe usar el guión bajo (underscore) "_".

Paquetes: Los nombres de los paquetes se escriben completamente en minúsculas.

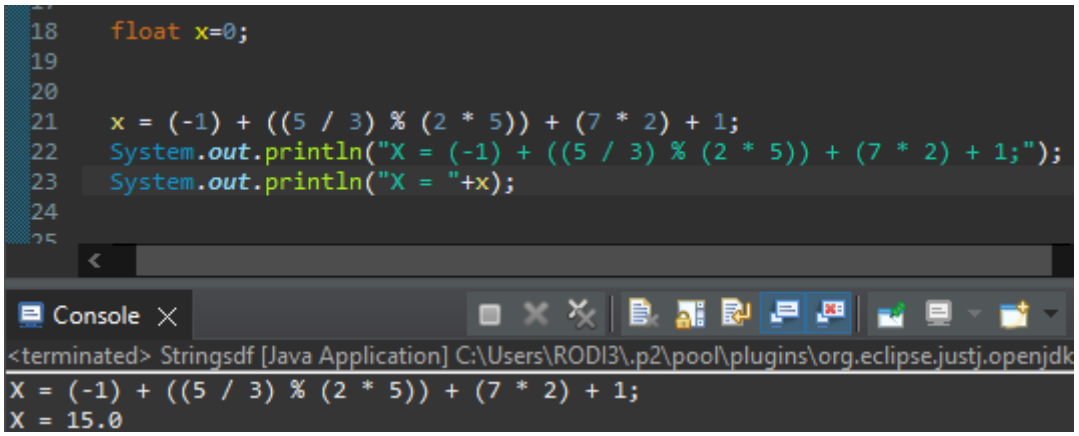
Estructuras de control: Cuando una sentencia forma parte de una estructura de control como por ejemplo de flujo como for o while, es necesario escribir los corchetes ({}), aunque solo se tenga una línea de código.

Espacios: Los espacios son necesarios para entender el código de mejor manera, para mantenerlo limpio, ordenado y distinguir el principio y el final de estructuras de control. Cada vez que se escribe algo dentro de una llave “{“, se hace luego de una tabulación.

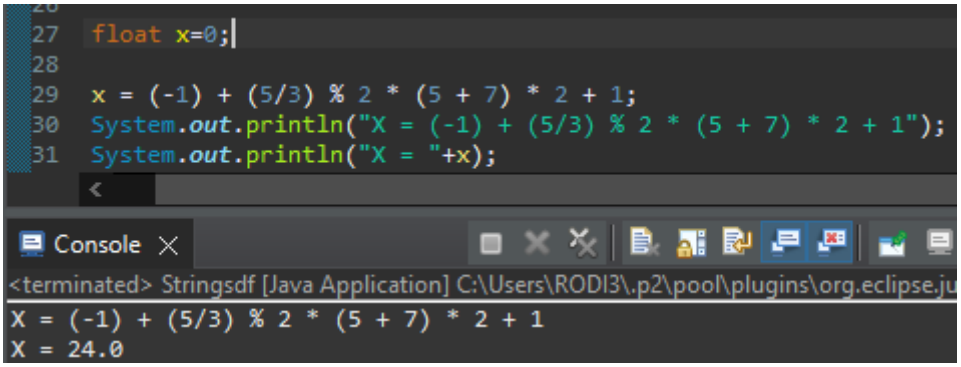
Comentarios: Los comentarios son útiles y recomendados para explicar ciertas partes del código o hacer aclaraciones sobre el mismo. Especialmente si es un proyecto en el que pueda trabajar más de un programador.

Ejercicios:

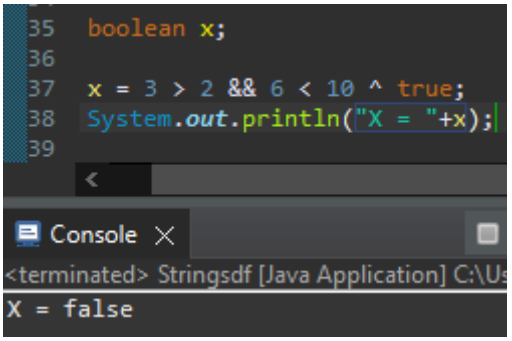
1) Indique el valor de x = -1 + 5 /3 % 2 * 5 + 7 * 2 + 1



2) Indique el valor de x = -1 + (5 /3) % 2 * (5 + 7) * 2 + 1



3) Indique el valor de x = 3 > 2 && 6 < 10 ^ true

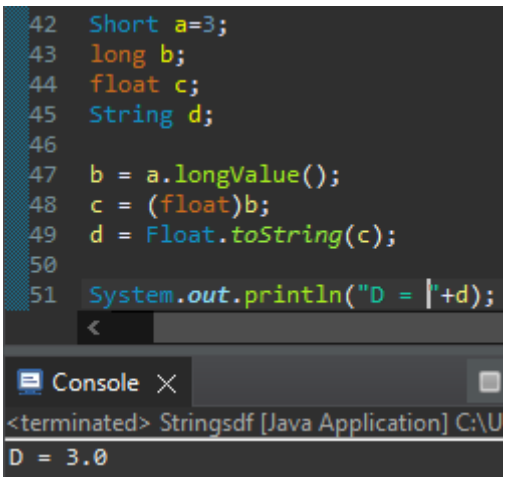


4) Se poseen 4 atributos:

- a de tipo short
- b de tipo long
- c de tipo float
- d de tipo String

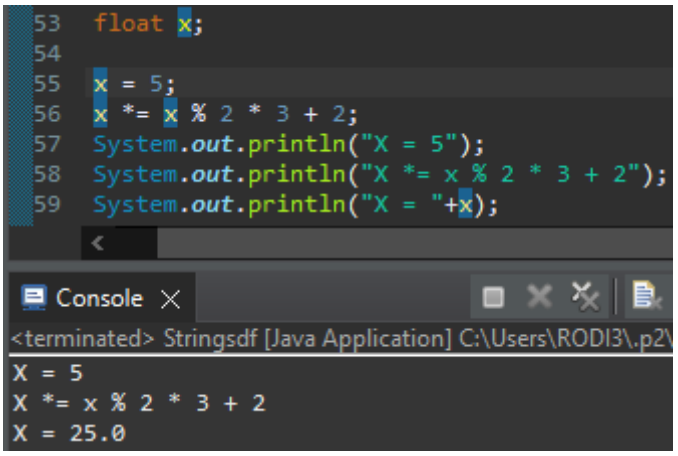
Escriba los pasos necesarios para guardar:

- a en b
- b en c
- c en d
- d en a



5) Indique el valor de que tendrá x si se tiene que:

1. x = 5;
2. x *= x % 2 * 3 + 2;



6) Indique el valor que tendrá x en cada una de las líneas:

1. x = 1; --------> **x** = 1
2. ++x; --------> **x** = 2
3. x += x++; --------> **x** = 4
4. --x; --------> **x** = 3

7) Indique el valor resultante de cada una de las líneas:

1. !(true ^ 10 > 5) & !(!true || !(false ^ false)) --------> **FALSE**
2. (10 < 5) && (false) || (false == true) ^ 8.3f >= 8.3f --------> **TRUE**
3. 5 < 11 ^ 5 > 15 | 2 == 2 & false ^ !true ^ 10 <= 10 --------> **TRUE**
4. !(true && true || false ^ !(4==3) & 3 > 1 ^ !true) --------> **FALSE**

8) Indique el valor resultante de cada una de las líneas

1. int i = 43; --------> **i** = 43
2. int mascara = 32; --------> **mascara** = 32
3. int bit = i & mascara; --------> **bit** = 32
4. bit >>= 5; --------> **bit** = 1
5. mascara ^= 1; --------> **mascara** = 33
6. bit |= i & mascara; --------> **bit** = 33
7. bit <<= 1; --------> **bit** = 66
8. mascara &= (~bit) >>> 3; --------> **mascara** = 33
9. bit += (~(~0)<<1); --------> **bit** = 66
10. mascara += bit + i; --------> **mascara** = 142

9) Leyendo los códigos correspondientes de la tabla de código ASCII descifre el mensaje:

```
1.  String mensaje;
2.  char caracter;
3.  int encriptado = 79 >> 2;
4.
5.  encriptado = (~((-160) >> 3) << 2) + 3;
6.  caracter = (char) encriptado;
7.  mensaje = String.valueOf(caracter);
8.
9.  encriptado = (((encriptado >> 1 ) ^ 30) << 1) -7;
10. caracter = (char) encriptado;
11. mensaje += String.valueOf(caracter);
```

El mensaje en código ASCII es: "***Ok***"

10) Si un color está representado por un entero en el cual, el byte más significativo corresponde al canal rojo, y los 3 bytes siguientes corresponden a los canales verde, azul y alfa respectivamente, genere un algoritmo que separe los valores rojo, verde, azul y alfa, y los guarde en cuatro atributos que correspondan a estos. Luego averigüe estos valores para el siguiente color:

2295136255

Para realizar este ejercicio debe utilizar operadores de bit.

```
int alfaByte=0, rojoByte=0, verdeByte=0, azulByte=0;
long num=2295136255l, bit=0;
int cont=0, cont2 = 0, posicion=0, colorByte=0;
String binario = "", alfa="", rojo="", verde="", azul="", numBin="";

while (num>0) {
    bit = num % 2;
    if (cont == 8) {
        binario += ".";
        cont = 0;
    }
    binario += Long.toString(bit);
    cont++;
    num = num / 2;
}
System.out.println(binario);

posicion=binario.indexOf(".");
alfa=binario.substring(0,posicion);
binario=binario.substring(posicion+1);

posicion=binario.indexOf(".");
azul=binario.substring(0,posicion);
binario=binario.substring(posicion+1);

posicion=binario.indexOf(".");
verde=binario.substring(0,posicion);
binario=binario.substring(posicion+1);

rojo=binario;
```

```java
for(int i = 0; i<alfa.length(); i++) {
    if(alfa.charAt (i) == '1') {
        alfaByte += Math.pow(2, i);
    }
}

for(int i = 0; i<rojo.length(); i++) {
    if(rojo.charAt (i) == '1') {
        rojoByte += Math.pow(2, i);
    }
}

for(int i = 0; i<verde.length(); i++) {
    if(verde.charAt (i) == '1') {
        verdeByte += Math.pow(2, i);
    }
}

for(int i = 0; i<azul.length(); i++) {
    if(azul.charAt (i) == '1') {
        azulByte += Math.pow(2, i);
    }
}
System.out.println("Azul: "+azulByte);
System.out.println("Verde: "+verdeByte);
System.out.println("Rojo: "+rojoByte);
System.out.println("Alfa: "+alfaByte);
```