Introducción a Ciencias de la Computación I

2015-1

Tarea 1

Profesor: José de Jesús Galaviz Casas

Ayudante: Luis Eduardo Castro Omaña

Ayudante: Karla Rocío Vargas Godoy

Alumno: Flores González Luis Brandon

8 de Septiembre del 2014

**1 Teoría**

Responde las siguientes preguntas:

1. Describe la arquitectura Von Neumann.

Se le conoce como arquitectura de una computadora a la organización que tienen en sus componentes electrónicos, y la manera como estos están integrados para funcionar.

En la arquitectura de von Neumann es una organización donde tenemos un procesador central en el que se ejecutan las operaciones aritméticas y de comparación (lógicas); una memoria central que se utiliza para almacenar datos, resultados intermedios y el programa a ejecutarse; tenemos unidades de entrada y salida (input/output) que sirven para darle a la computadora el programa y los datos y recibir los resultados; por último, tenemos memoria externa o auxiliar, como discos, diskettes, cintas magnéticas, que nos sirven para almacenar, ya sean datos o programas, de una ejecución a otra, sin tener que volver a realizar el proceso, o sin que tengamos que volverlos a proporcionar.

1. Explica la diferencia que existe de un programa compilado a uno interpretado.

Un intérprete es un programa que una vez cargado en la memoria de una computadora, y al ejecutarse procede como sigue:

-Toma un enunciado del programa en lenguaje de alto nivel, llamado el código fuente.

-Traduce ese enunciado y lo ejecuta.

-Repite estas dos acciones hasta que alguna instrucción le indique que pare, o bien, tenga un error fatal en la ejecución.

Un compilador es un programa que una vez que reside en la memoria, y al ejecutarse toma un programa fuente y lo traduce completo a un programa equivalente en otro lenguaje de programación, que generalmente es lenguaje de máquina.

Mientras que un intérprete va traduciendo y ejecutando, el compilador no se encarga de ejecutar, sino simplemente de producir un programa equivalente, susceptible de ser cargado a la memoria de la máquina y ser ejecutado en un momento posterior.

A los intérpretes se les conoce también como máquinas virtuales, porque una vez que están cargados en una máquina se comportan como si fueran otra computadora, aquella cuyo lenguaje de máquina es el que se está traduciendo y ejecutando.

1. Explica brevemente las acciones que realiza el compilador de Java durante el proceso de compilación.

Java es un lenguaje orientado a objetos, cuyo principal objetivo de diseño es su portabilidad, esto es que sin cambios se pueda ejecutar en cualquier computadora. Una marea de hacer programas escritos en Java es mediante lo siguiente:

-Se traduce el programa escrito en Java a un lenguaje de bajo nivel, tipo lenguaje de máquina, pero que no sea el de una máquina en específico.

-Se construye (programa) un intérprete de este “lenguaje de máquina”, conocido como una máquina virtual, y se ejecuta el programa en lenguaje de máquina virtual de Java.

El “lenguaje de máquina” de Java se llama bytecode. Es más fácil construir una máquina virtual que entienda el bytecode que construir un compilador para cada posible lenguaje de máquina. Además, una vez que está definida la máquina virtual se le pueden agregar capacidades al lenguaje simplemente dando su transformación a bytecode.

Por todo esto, para ejecutar un programa escrito en Java necesitamos:

a) Tener un compilador de Java que traduzca de programas escritos en Java a bytecode (javac)

b) Tener un intérprete de bytecode (o máquina virtual de Java) a la que se le da como datos el programa en bytecode y los datos pertinentes al programa.

1. ¿Qué es el bytecode?

El bytecode es un código intermedio más abstracto que el código máquina. Habitualmente se lo trata como a un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario que contiene código máquina producido por el compilador.   
El bytecode recibe su nombre porque generalmente cada código de operación tiene una longitud de un byte, si bien la longitud del código de las instrucciones varía. Cada instrucción tiene un código de operación entre 0 y 255 seguido de parámetros tales como los registros o las direcciones de memoria. Esta sería la descripción de un caso típico, si bien la especificación del bytecode depende ampliamente del lenguaje. 

1. ¿Qué es un paradigma de programación? Da unos ejemplos.

Un paradigma de programación provee (y determina) la visión y métodos de un programador en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas (con la solución de múltiples “problemas” se construye una aplicación).

Los lenguajes de programación son basados en uno o más paradigmas Por ejemplo:

Smalltalk y Java son lenguajes basados en el paradigma orientado a objetos. El lenguaje de programación Scheme, en cambio, soporta sólo programación funcional. En cambio Python, soporta múltiples paradigmas.

1. ¿Qué es POO? Menciona 5 conceptos fundamentales de POO y explícalos.

La programación orientada a objetos es un programa con un conjunto de objetos, que se comunican entre ellos para realizar tareas y que es un modelo que representa un subconjunto del mundo real, tan fielmente como sea posible , de modo más fácil y natural, donde los objetos van a tener características (atributos) y comportamientos (métodos). Que a diferencia de los lenguajes procedurales, en donde los datos y los procedimientos se encuentran separados y sin relación alguna.

Los lenguajes procedurales, utilizan funciones y después les pasan a datos en tanto que los lenguajes orientados a objetos definen objetos y después envían mensajes a los objetos diciendo que realicen alguno de los métodos especificados para el objeto.

Entre las ventajas de la programación orientada a objetos es que los métodos están pensados para hacer programas y módulos más fáciles de escribir, mantener y reutilizar, así como que sean modulares y reutilizables parte de los códigos de estos programas

5 Conceptos fundamentales de POO son:

Atributos: Algunos sistemas o lenguajes orientados a objetos llaman a estos variables propias, variables miembro o variables de estado. Se refieren a los valores que debe recordar cada objeto y que describe el estado del objeto. Cada objeto puede tener un valor distinto en cada uno de estos atributos. Como uno de sus nombres indica (variables de estado), conforman el estado del objeto, que puede cambiar durante la ejecución de la aplicación.

Mensajes (messages): Un objeto le pide un servicio a otro mandándole un mensaje. A esta acción le llamamos el envió de un mensaje y es la única manera en que un objeto se puede comunicar con otro. Un mensaje consiste del nombre de una operación y los argumentos que la operación requiere.

Comportamiento o conducta (behaviour): El conjunto de mensajes a los que un objeto puede responder es lo que se conoce como la conducta o el comportamiento del objeto. Al nombre de la operación en el mensaje le llamamos el nombre del mensaje.

Métodos (methods): Cuando un objeto recibe un mensaje, ejecuta la operación que se le solicita mediante la ejecución de un método. Un método es un algoritmo paso a paso que se ejecuta como respuesta a un mensaje cuyo nombre es el mismo que el del método. Para que un método pueda ser invocado desde n objeto de otra clase, debe ser público. En el caso de algunos lenguajes de programación, a los métodos se les llama funciones miembro, porque son funciones (procedimientos) que están definidos en, son miembros de, la clase.

Clases (classes): Si dos objetos presentan el mismo comportamiento, decimos que pertenecen a la misma clase. Una clase es una especificación genérica para un número arbitrario de objetos similares. Las clases permiten construir una taxonomía de los objetos en un nivel abstracto, conceptual. Nos permiten describir a un grupo de objetos. Por ejemplo, cuando describimos a los seres humanos damos aquellas características que tienen en común. Cada ser humano es un objeto que pertenece a esa clase. Hay que insistir en que las clases corresponden únicamente a descripciones de objetos, no tienen existencia en sí mismas.

1. Explica que es la clase Object

La clase **Object**, como ya se ha indicado anteriormente, es la clase raíz de todo el árbol de la jerarquía de clases Java, y proporciona un cierto número de métodos de utilidad general que pueden utilizar todos los objetos. La lista completa se puede ver en la documentación del API de Java, aquí solamente se tratarán algunos de ellos; por ejemplo, **Object** proporciona:

* Un método por el que un objeto se puede comparar con otro objeto
* Un método para convertir un objeto a una cadena
* Un método para esperar a que ocurra una determinada condición
* Un método para notificar a otros objetos que una condición ha cambiado
* Un método para devolver la clase de un objeto

1. ¿Cuál es el entero positivo con signo más grande que cabe en 32 bits (o sea un int)? ¿Qué ocurre cuando le sumas uno a ese valor en una variable int?

El entero positivo con signo más grande que cabe en 32 bits es 2 147 483 647. Y cuando le sumas uno a ese valor se regresa al número mínimo negativo que será -2 147 483 648

1. ¿Por qué preferimos hacer que los atributos de un objeto sean "private"?

Así sólo se permite el acceso a objetos de la misma clase además sirve para que solo nosotros podamos ver los atributos de ese objeto.

10. Explica cómo es la representación del tipo doublé en la máquina.

**2 Ejercicios**

1. ¿Cuál será el valor en pantalla de los siguientes códigos:

(a) public class Tarea {

public static void main(String [] args){

int a = 10,b = 0,c = 4;

//b = a\*b(a-b)+c;

b = a\*b\*(a-b)+c;

/\*

b = -100;

a = (b \* 2)/c;

\*/

c = ((-1\*(b - 9)) % 2) + a;

System.out.println(c);

}

}

El valor en pantalla es 11

(b) public class Tarea {

public static void main(String [] args){

int c = -1+5\*7-12/3;

System.out.println(c);

}

}

El valor en pantalla es 30

(c) public class Tarea {

public static void main(String [] args){

int c = (-1+5)\*7-(12/3);

System.out.println(c);

}

}

El valor en pantalla es 24

2. ¿Cuál es el nombre que no es mostrado en pantalla?

public class Tarea {

public void ejercicio(int n){

if(n >=5){

switch(n){

case 5:

case 6:

n += 1;

System.out.println("Karla");

case 7:

n = 20;

break;

case 8:

System.out.println("Emmanuel");

break;

case 9:

System.out.println("Eduardo");

case 10:

System.out.println("Galaviz");

default:

System.out.println("Goku");

}

}

else{

System.out.println("Roberto");

}

}

public static void main(String [] args){

new Tarea().ejercicio(1);

new Tarea().ejercicio(5);

new Tarea().ejercicio(7);

new Tarea().ejercicio(8);

new Tarea().ejercicio(10);

}

}

El nombre que no es mostrado en pantalla es “Eduardo”

3. ¿Qué números se imprimen con el siguiente código?

Explica cuántas iteraciones son realizadas en el for

public class Tarea {

public static void main(String [] args){

for(int i = 1; i <= 1000;i++){

if(i%2==0){

continue;

}

if(i%3==0){

continue;

}

System.out.println(i);

if(i >= 15){

break;

}

}

}

}

Se imprimen los números 1, 5, 7, 11, 13, 17. Y se hacen 999 iteraciones.

4. Realiza el código para calcular el factorial de un número (Hint: usa ciclos)

n! = n(n-1)(n-2)....(1)

public class Matematicas{ //Clase Matematicas para calcular factorial de un numero

public static long getFactorial(long ln){

if(ln == 0){

return 1;

}

if(ln<0){

return -1;

}

return getFactorial(ln, ln);

}

public static long getFactorial(long ln, long last){//last ultimo num y ln el num que se calcula

System.out.println("Acumulado: "+ln); //Numero acumulado

System.out.println("Numero actual: "+last); //Numero actual

if(last == 1){ //

return ln;

}

else{

return getFactorial(ln\*(last-1), last-1);

}

}

public static void main(String[] args){

System.out.println(getFactorial(4)); // Calcula el factorial del numero

}

}// El resultado que muestra en pantalla es 24

5. Con la clase Punto vista en clase, realiza una clase Triangulo, que tenga como atributos Puntos, que de la clase puedas obtener el área de los triángulos, las distancias de sus lados, el perímetro y saber si el triángulo es equilátero, isósceles o escaleno.

(Hint: verificar en el constructor que los puntos sí crean un triángulo).