Examen Parcial de P.O.O – Grupo 01

1) **Describe como mínimo 3 paradigmas de programación y dar sus características.**

a) Paradigma estructurado:

• Se usa para una secuencia de decisión y repetición.

• Es un código más fácil de leer, pero todavía algo difícil para sistemas grandes debido a repetición de código.

b) Paradigma procedural o procedimental:

• Uso de subprogramacion

– Agrupamiento de código permitiendo la creación de acciones complejas.

– Atribución de un nombre para esas acciones complejas.

– Llamada a esas acciones complejas de cualquier punto de programa.

• Esas acciones complejas son denominadas procedimientos, subrutinas y funciones

c) Paradigma Orientado a Objetos:

• Clases de objetos

– Agrupamiento de procedimientos y variables afines.

• Pacotes de clases

– Agrupamiento de clases afines.

– Representan bibliotecas de apoyo.

2) **Cual es la diferencia entre i++ e ++i, dar un ejemplo con un código.**

Manera general de uso de operadores de incremento de decremento

++ Preincremento (++j): Incrementar "i" en 1, después utilizar el nuevo valor de "i" en la expresión en que esta variable reside.

Postincremento++ (i++): Usar el valor actual de "i" en la expresión en la que esta variable reside, después incrementar a en 1.

--Predecremento (--i): Decrementar "i" en 1, después utilizar el nuevo valor de "i" en

la expresión en que esta variable reside.

Postdecremento-- (i--): Usar el valor actual de "i" en la expresión en la que esta variable reside, después decrementar "i" en 1.

Ejemplos a manera de códigos:

```
public class incredecremento {
       // Operadores de preincremento y postincremento.

        public static void main( String args[] ){
        int c ;
        // demostración del operador de postincremento y declaro la
variable in
        c = 5; // asigno 5 a c

        System.out.println( c ); // imprime 5
        System.out.println( c++ ); // imprime 5, después postincrementa
        System.out.println( c ); // imprime 6 , la nueva variable

        System.out.println(); // omite una línea en la consola de java

        // demostración del operador preincremento y declaro la variable
        c = 5; // asigno 5 a c

        System.out.println( c ); // imprime 5
        System.out.println( ++c ); // preincrementa y después imprime 6
        System.out.println( c ); // imprime 6 , ósea el nuevo valor
directamente

        }

        }
```

3) Completar el cuadro con la información de PRIORIDAD, siendo el caso que sea 1 más prioritario que 5.

Operador	Prioridad	Operador	Prioridad
/	**3**	(expr)	**1**
--var	**2**	+expr	**2**
*	**3**	+	**4**
%	**3**	-	**4**
Var--	**2**	&&	**5**

4)Responder las siguientes preguntas y dar ejemplos:

a) ¿Que significa un casting en programación?

En programación, un casting (o cast) sirve para cambiar el tipo de dato del valor resultante de una expresión (manera simple).

De manera más formal este es un procedimiento para transformar una variable primitiva de un tipo a otro y también para transformar un objeto de una clase a otra clase (debe haber una relación de herencia entre estas).

Casting implícito	**Casting explícito**

_No se necesita escribir código para que la conversión.
 	
_Se hace una conversión ancha (widening casting), colocar un valor pequeño en un contenedor grande.
 	
EJEMPLO:
 	
int n1 = 100;
 	
long n2 = n1; // int cabe en un long
 	
long n2 = 100; // 100 en un int
	_Sí es necesario escribir código.
	
	_Realiza una conversión estrecha (narrowing casting), colocar un valor grande en un contenedor pequeño, aumentando la susceptibilidad de perder de datos.
	
	EJEMPLO:
	
	int num1 = 100;
	
	short num2 = (short) num1; // Aquí hace falta un casting explícito: short tiene menor rango que int.
	
	(tipo) valor_a_convertir.

tener en cuenta lo siguiente:

_No es posible realizar casting entre una variable primitiva booleana y cualquier otra variable primitiva.

_ long > int > short > byte.

_Sí es posible realizar casting entre una variable primitiva char y una variable primitiva que almacene enteros:

```
int num1 = 164;

char letra = (char) num1;

System.out.println(letra);

System.out.println((char) 164);
```

b)¿Qué es una función y un procedimiento?

Funcion:

Una función es una parte del programa que puede manipular datos y devolver un valor de un cierto tipo. Osea es como un proceso que recibe valores de entrada (llamados parámetros) y el cual retorna un valor resultado. Adicionalmente, las funciones son subprogramas dentro de un programa, que se pueden invocar (ejecutar) desde cualquier parte del programa, es decir, desde otra función, desde la misma función o desde el programa principal, cuantas veces sea necesario.
A manera de pseudocódigo:

```
función <nombre> (param1 : tipo1 ,parm2 : tipo2 ..., paramN : tipo N) : tipo
variables
        <declaraciones>
inicio
        <instrucciones>
        retornar <expresión>
fin_funcion
```

EJEMPLO: La función minimo que en matemáticas se define como sigue:

minimo: Reales x Reales x Reales ⇒ Reales

minimo(a , b , c) = a ,si a <= b y a<= c

minimo(a , b , c) = = b ,si b<=a y b<=c

minimo(a , b , c) = = c ,si c<=a y c<=b

```
funcion minimo( a : real, b : real , c : real ): real
inicio
 if (a <=b & a <=c) then
 write (“a”);
 else
 if (b <=a & b<=c) then
 write(“b”)
else
write(“b”)
end_if
end_funcion
```

Procedimiento:

Un procedimiento se puede asimilar a una función que puede retornar más de un valor mediante el uso de parámetros por referencia y se usan para evitar duplicación de código y conseguir programas más cortos, también una herramienta conceptual para dividir un problema en subproblemas (divide y vencerás) logrando de esta forma escribir más fácilmente programas grandes y complejos.

A manera de pseudocódigo:

```
procedimiento <nombre> ( param1: tipo1, ..., paramn : tipon)
variables
        <declaraciones>
inicio
        <instrucciones>
fin_procedimiento
```

EJEMPLO: Desarrollar un procedimiento que intercambie los valores de dos variables enteras, es decir, que implemente el intercambio para variables enteras.

```
procedimiento intercambio (ref x : entero, ref y : entero)
variables   /* variable auxiliar para realizar el intercambio */
 aux : entero
inicio
 aux: =x            /* se almacena el valor de x en la variable aux */
 x :=y              /* se almacena el valor de y en la variable x */
 y :=aux            /* se almacena el valor original de x en la variable y */
 fin_procedimiento
```

c)¿Qué quiere decir sobrecarga de métodos ?

Explicación: La sobrecarga de métodos en el tema de P.O.O quiere decir a la posibilidad de tener 2 o más métodos que posean el mismo nombre, pero con funcionalidad distinta, ósea 2 o más métodos de un mismo nombre realizando acciones diferentes de tal manera que el compilador usara una o la otra dependiendo de los parámetros comprometidos y por eso se puede diferenciar ya sea por cantidad, tipo u orden. También hace que un mismo nombre pueda representar distintos métodos con distinto tipo y número de parámetros, manejados dentro de la misma clase.

_Con ello una clase puede tener distinto comportamiento dependiendo de cual método sobrecargado se utilice, a esta característica se le conoce como Polimorfismo por sobrecarga.

_Como ejemplos se toman los ejercicios que tengan que ver con polimorfismo ya que esta tiene una clase de variar su comportamiento, en su caso por sobrecarga, el cambio del comportamiento de una clase se define sobrecargando los métodos necesarios para lograr el polimorfismo requerido.

c,1)¿Qué quiere decir sobrecarga de operadores?

La sobrecarga de operadores es la capacidad para transformar los operadores de un lenguaje como por ejemplo el +, -, etc, cuando se dice transformar se refiere a que los operandos que entran en juego no tienen que ser los que admite el lenguaje por defecto. Mediante esta técnica podemos sumar dos objetos creados por nosotros o un objeto y un entero, en vez de limitarnos a sumar números enteros o reales, por ejemplo.

La sobrecarga de operadores ya era posible en c++ y en otros lenguajes, pero sorprendentemente java no lo incorpora, así que podemos decir que esta característica es una ventaja de c++ respecto a java, aunque mucha gente, esta posibilidad, no lo considera una ventaja porque complica el código.

A la hora de hablar de operadores vamos a distinguir entre dos tipos, los unarios y los binarios. Los unarios son aquellos que solo requieren un operando, por ejemplo a++, en este caso el operando es 'a' y el operador '++'. Los operadores binarios son aquellos que necesitan dos operadores, por ejemplo a+c , ahora el operador es '+' y los operandos 'a' y 'c'. Es importante esta distinción ya que la programación se hará de forma diferente.

Los operadores que podemos sobrecargar son los unarios, +, -, !, ~, ++, --; y los binarios +, -, *, /, %, &, |, ^, <<, >>. Es importante decir que los operadores de comparación, ==, !=, <, >, <=, >=, se pueden sobrecargar pero con la condición que siempre se sobrecargue el complementario, es decir, si sobrecargamos el == debemos sobrecargar el !=.