

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ FACULTAD
DE INGENIERIA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE PROGRAMACIÓN DE COMPUTADORAS

DESARROLLO DE SOFTWARE II
1LS116

PROFESOR:
ING. JOSÉ JAVIER CHIRÚ F.

II SEMESTRE
2022



ESTRUCTURAS DE CONTROL

MÓDULO 4

Instrucciones de alternativa

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

La estructura alternativa permite bifurcar el “flujo” del programa en función de una expresión lógica; disponemos de tres estructuras alternativas diferentes: alternativa simple, alternativa doble y alternativa múltiple.

Alternativa Simple

```
int valor =10;  
if (valor<20){  
    JOptionPane.showMessageDialog( parentComponent: null, message: "Verdadero");  
}
```

Alternativa Double

```
int valor =10;
if (valor<20){
    JOptionPane.showMessageDialog( parentComponent: null, message: "Verdadero");
}else{
    JOptionPane.showMessageDialog( parentComponent: null, message: "Falso");
}
```

Alternativa Multiple

```
int valor =10;
if (valor>=10){
    JOptionPane.showMessageDialog( parentComponent: null, message: "Mayor");
}else if (valor == 0){
    JOptionPane.showMessageDialog( parentComponent: null, message: "Igual");
}else {
    JOptionPane.showMessageDialog( parentComponent: null, message: "Menor");
}
```

Comparar String

```
String cadenapruebas="EGPH";  
if (cadenapruebas.equals("EGPH")){  
    // Ambas cadenas son iguales (no es el caso)  
}else {  
    // Ambas cadenas no son iguales (es el caso)  
}
```

Comparar String

```
String cadenapruebas="EGPH";  
if (cadenapruebas.equalsIgnoreCase( anotherString: "EGPH")){  
    //ignora si los caracteres están compuestos por minúsculas o mayúsculas  
}else {  
    //Ambas cadenas no son iguales (es el caso)  
}
```


Switch Case


La instrucción switch es una instrucción de múltiples vías. Proporciona una forma sencilla de enviar la ejecución a diferentes partes del código en función del valor de la expresión. Básicamente, la expresión puede ser tipos de datos primitivos byte, short, char e int

Switch Case

```
int day = 5;
String dayString;
// instrucción switch con tipo de datos int
switch (day)
{
    case 1: dayString = "Lunes";
            break;
    case 2: dayString = "Martes";
            break;
    case 3: dayString = "Miercoles";
            break;
    case 4: dayString = "Jueves";
            break;
    case 5: dayString = "Viernes";
            break;
    case 6: dayString = "Sabado";
            break;
    case 7: dayString = "Domingo";
            break;
    default: dayString = "Dia inválido";
            break;
}
```

Multiples cases

```
int day = 5;
String dayType;
switch (day)
{
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        dayType = "Dia laborable";
        break;
    case 6:
    case 7:
        dayType = "Fin de semana";
        break;
    default: dayType= "Tipo de dia invalido";
}
```



Sentencias Repetitivas o Ciclos Repetitivos

Sentencias Repetitivas o Ciclos Repetitivas

Las estructuras repetitivas también llamadas ciclos permiten repetir una secuencia de instrucciones un número determinado de veces, al hecho de repetir la ejecución de una secuencia de acciones se le llama iteración

For

Un for en programación se usa cuando queremos repetir un conjunto de instrucciones un número finito de veces.

For

```
for(int i = valor inicial; i <= valor final; i = i + paso)
{
    ....
    ....
    Bloque de Instrucciones....
    ....
    ....
}
```

Ejemplo de For par Calculando un Factorial

```
double factnumero =1;  
int contador;  
for (contador=1; contador <= num; contador++){  
    factnumero*=contador;  
}
```


Ejemplo de un For en Clase Secundaria

```
public class Factorial {  
    private int num;  
  
    public void asignar(int numcalcular){  
        num = numcalcular;  
    }  
  
    public double calcularFactorial() {  
        double factnumero = 1;  
        int contador;  
        for (contador=1; contador <= num; contador++){  
            factnumero*=contador;  
        }  
        return factnumero;  
    }  
}
```

Ejemplo de un For en Clase Principal

```
public class MainPunto2 {  
    public static void main(String[] args) {  
        int num, contador;  
        double resp;  
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
        Factorial objFact = new Factorial();  
        try{  
            System.out.println("Ingrese un numero entero Positivo ");  
            num=Integer.parseInt(br.readLine());  
            if (num>=0){  
                for(contador=0;contador<=num;contador++){  
                    objFact.asignar(contador);  
                    resp= objFact.calcularFactorial();  
                    System.out.println(contador+"!="+ resp);  
                }  
            }else System.out.println("El valor ingresado no es positivo");  
        }catch (Exception ex){  
            System.out.println("Ocurrio un Error");  
        }  
    }  
}
```

while

while se va repitiendo el código en base a una condición, es decir, mientras esa condición sea verdadera.

while

```
while(condicion) {
```

```
....
```

```
....
```

```
Bloque de Instrucciones....
```

```
....
```

```
....
```

```
}
```

Ejemplo de While Calculando un Factorial

```
double factnumero =1;  
int contador=1;  
while(contador <= num ){  
    factnumero*=contador;//Acumulador  
    contador++;//Contador  
}  
return factnumero;
```

Ejemplo de un While en Clase Secundaria

```
public class Factorial {  
    private int num;  
  
    public void asignar(int numcalcular){  
        num = numcalcular;  
    }  
  
    public double calcularFactorial(){  
        double factnumero = 1;  
        int contador = 1;  
        while (contador <= num){  
            factnumero *= contador; // Acumulador  
            contador++; // Contador  
        }  
        return factnumero;  
    }  
}
```

Ejemplo de un While en Clase Principal

```
public class MainPunto2 {  
    public static void main(String[] args) {  
        int num, contador=0;  
        double resp;  
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
        Factorial objFact = new Factorial();  
        try{  
            System.out.println("Ingrese un numero entero Positivo ");  
            num=Integer.parseInt(br.readLine());  
            if (num>=0){  
                while(contador<=num){  
                    objFact.asignar(contador);  
                    resp= objFact.calcularFactorial();  
                    System.out.println(contador+"!="+ resp);  
                    contador++;  
                }  
            }else System.out.println("El valor ingresado no es positivo");  
        }catch (Exception ex){  
            System.out.println("Ocurrio un Error");  
        }  
    }  
}
```

Do while

Do while revisa la condición al final; por eso es el único que por lo menos una vez realiza una interacción si la condición a evaluar es falsa.

do while

```
do {
```

```
....
```

```
....
```

```
Bloque de Instrucciones....
```

```
....
```

```
....
```

```
} while(condicion) ;
```

Ejemplo de do While Calculando un Factorial

```
double factnumero =1;  
int contador=1;  
do {  
    factnumero*=contador; //Acumulador  
    contador++; //Contador  
}while(contador <= num );
```

Ejemplo de un do while en Clase Secundaria

```
public class Factorial {  
    private int num;  
  
    public void asignar(int numcalcular){  
        num = numcalcular;  
    }  
  
    public double calcularFactorial(){  
        double factnumero = 1;  
        int contador = 1;  
        do {  
            factnumero *= contador; // Acumulador  
            contador++; // Contador  
        } while (contador <= num);  
        return factnumero;  
    }  
}
```

Ejemplo de un do While en Clase Principal

```
public class MainPunto2 {  
    public static void main(String[] args) {  
        int num, contador=0;  
        double resp;  
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
        Factorial objFact = new Factorial();  
        try{  
            System.out.println("Ingrese un numero entero Positivo ");  
            num=Integer.parseInt(br.readLine());  
            if (num>=0){  
                do{  
                    objFact.asignar(contador);  
                    resp= objFact.calcularFactorial();  
                    System.out.println(contador+"!="+ resp);  
                    contador++;  
                }while(contador<=num);  
            }else System.out.println("El valor ingresado no es positivo");  
        }catch (Exception ex){  
            System.out.println("Ocurrio un Error");  
        }  
    }  
}
```

SENTENCIAS BREAK

La sentencia break hace que se salga del bucle inmediatamente por lo que no se ejecutara ni el código que se encuentre después del break en esa misma iteración ni ninguna de las posibles iteraciones restantes.