Ingeniería en Sistemas Computacionales

Fundamentos de Programación

Clave AED-1285

SATCA 2-3-5

Unidad 3. Control de flujo

Dra. María Lucía Barrón *Estrada*

Fechas de Examen/Evaluación Agosto-Diciembre 2018

- ~~Unidad 1- 21 septiembre~~
- Unidad 2- 22 octubre
- Unidad 3- 23 noviembre
- Unidad 4- 7 diciembre
- Unidad 5- 7 diciembre

3. Control de flujo

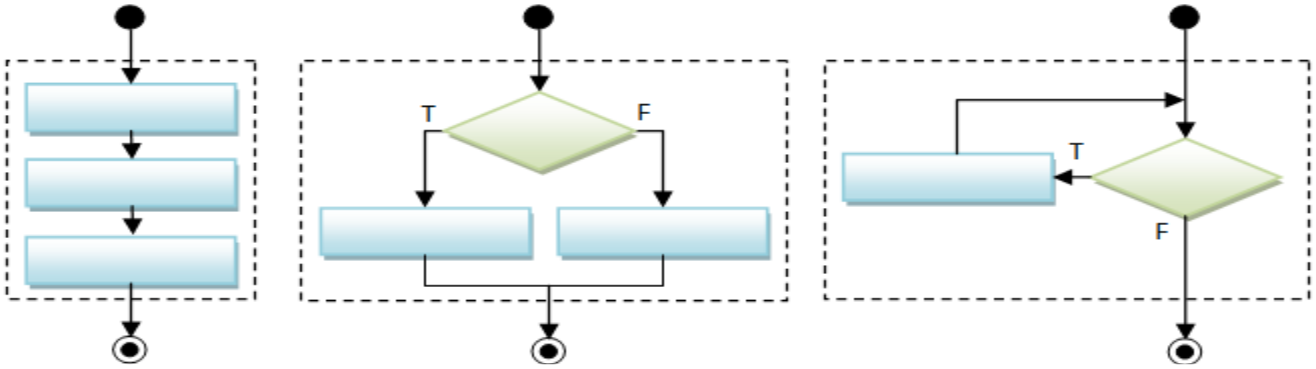
Conoce y aplica las estructuras condicionales y repetitivas de un lenguaje de programación para resolver problemas reales.

3.1 Estructuras secuenciales.

3.2 Estructuras selectivas: simple, doble y múltiple.

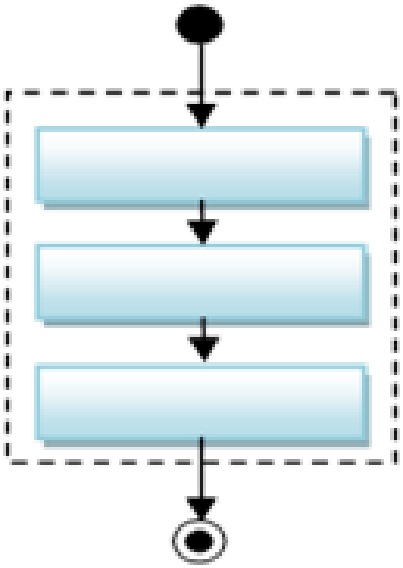
3.3 Estructuras iterativas: repetir mientras, hasta, desde

3.4 Diseño e implementación de funciones



3.1 Estructuras secuenciales.

- Estatutos simples
  - Asignación
  - Incremento
  - break
  - Llamada a métodos o funciones
- Bloques de instrucciones
  - { }



Secuencial

Instrucciones (Sentencias)

Una instrucción forma una completa unidad de ejecución.

Los siguientes tipos de expresiones pueden ser convertidas en instrucciones finalizando la expresión con un punto y coma, (;).

1. Expresiones de asignación	cont = 0;
2. Cualquier uso de ++ ó --.	totalMujeres++;
3. llamadas a métodos.	Math.pow(x,2);
4. Expresiones de creación de objetos.	new String();

Bloques

Un **bloque** es un grupo de cero o más instrucciones entre llaves balanceadas (que abren y cierran) y se utilizan para agrupar instrucciones.

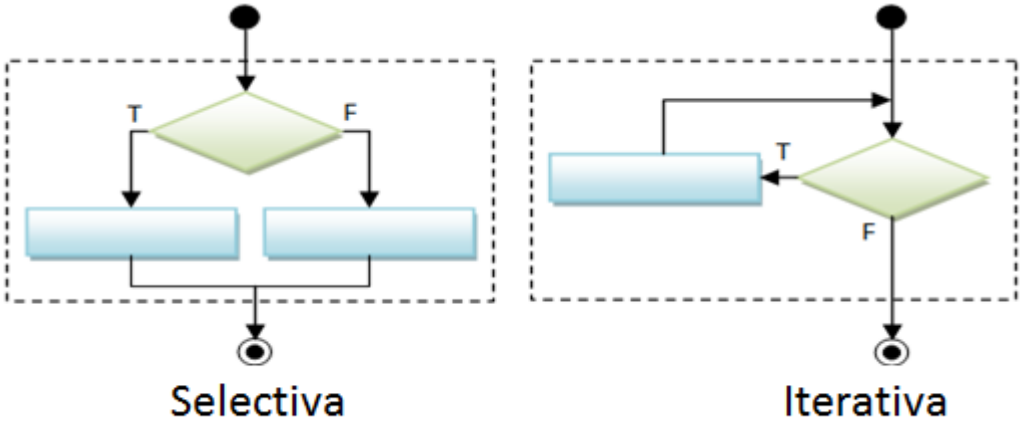
El listado siguiente muestra dos bloques del programa MaxVariablesDemo, conteniendo una instrucción simple:

```
aChar = ‘x’;
if (Character.isUpperCase(aChar)) {
    System.out.println(("Respuesta verdadera");
    System.out.println("El caracter " + aChar + " es mayúscula");
}
System.out.println ("Continua la ejecución");
```

Instrucciones de Control de Flujo

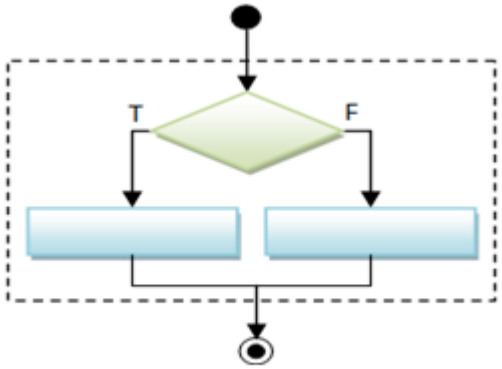
Se pueden emplear instrucciones de control de flujo para:

1. **Condicionalmente** ejecutar instrucciones,
2. **Cambiar el control secuencial** del flujo.
3. **Repetidamente** ejecutar bloques de instrucciones



3.2 Estructuras selectivas: simple, doble y múltiple.

- Selección simple (if)
- Selección doble (if – else)
- Selección múltiple (switch)



Selectiva

If/Else

La instrucción if / else permite al programa ejecutar selectivamente otras instrucciones, basando la selección en el resultado de una expresión booleana.

La forma general sería:

Condición simple

if (expression) statement

Condición doble

if (expression) statement1 **else** statement2

Ej: IfElseDemo.java

```
class IfElseDemo {
    public static void main(String[] args) {

        int testscore = 76;
        char grade;

        if (testscore >= 90) {
            grade = 'A';
        } else if (testscore >= 80) {
            grade = 'B';
        } else if (testscore >= 70) {
            grade = 'C';
        } else if (testscore >= 60) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        System.out.println("Grade = " + grade);
    }
}
```

Ejemplos usando if

```
if (height <= MAX) adjustment = 0;
else
    adjustment = MAX-height;

if (total < 7) System.out.println("Total menor que 7");

if (firstCh != 'a'  &&  limit < MAX) count++;
else
    count = count /2;
```

La Instrucción Switch

La instrucción **switch** se emplea para ejecutar instrucciones selectivamente, y la selección depende de el valor que arroja la expresion. El tipo de dato de la expresion puede ser primitivo byte, short, char, e int o referencia String, enumeración, Character, Byte, Short, and Integer.

Ej: SwitchDemo.java

```
switch (expresion) {
  case expresion :  estatutos;
                    break;
  case expresion :  estatutos;
                    break;
  // mas casos

  default : estatutos
}
```

- La instrucción **switch** evalúa su expresión, y ejecuta la instrucción (case) adecuada.
- La expresión debe ser de tipo primitivo (**char, byte, short o int)** o referencia **(String, Character, Byte, Short, Integer o Enumerado)**.
- La expresión de cada caso, debe ser **del mismo tipo** de la expresión del switch.
- La expresión de cada caso debe ser **ÚNICA**.
- La sección **default** es opcional y se ejecuta cuando ningún caso coincide con la expresión.
- Instrucción **break** después de cada caso (case). Cada instrucción break termina el bloque de switch, y el flujo de control continúa con la primera instrucción que sigue después del bloque de switch. Si no se colocaran instrucciones break, el cursor de ejecución seguiría ejecutando los siguientes bloques case hasta terminar o encontrar un break.

```
class SwitchDemo {
    public static void main(String[] args) {
        int month = 8;
        switch (month) {
            case 1:  System.out.println("January"); break;
            case 2:  System.out.println("February"); break;
            case 3:  System.out.println("March"); break;
            case 4:  System.out.println("April"); break;
            case 5:  System.out.println("May"); break;
            case 6:  System.out.println("June"); break;
            case 7:  System.out.println("July"); break;
            case 8:  System.out.println("August"); break;
            case 9:  System.out.println("September"); break;
            case 10: System.out.println("October"); break;
            case 11: System.out.println("November"); break;
            case 12: System.out.println("December"); break;
            default: System.out.println("Invalid month.");break;
        }
    }
}
```

```
class SwitchDemo2 {
   public static void main(String[] args) {
      int month = 2;
      int year = 2000;
      int numDays = 0;
      switch (month) {
         case 1:    case 3:    case 5:    case 7:  case 8:    case 10:
         case 12:
            numDays = 31;
            break;
         case 4:   case 6:    case 9:   case 11:
            numDays = 30;
            break;
         case 2:
            if ( ((year % 4 == 0) && !(year % 100 == 0))
                 || (year % 400 == 0) )
               numDays = 29;
            else
               numDays = 28;
            break;
         default:
            System.out.println("Invalid month.");
            break;
      }
      System.out.println("Number of Days = " + numDays);
   }
}
```

Ejemplo usando switch

```
char opcion;
System.out.println("Selecciona la opción deseada (A o B)");
opcion =  Keyboard.readChar();
switch (opcion){
  case 'A' : System.out.println("Seleccionaste la opción A");
             break;
  case 'b' :  // acepta b o B
  case 'B' : System.out.println("Seleccionaste la opción B");
             break;
  default : System.out.println("Opcion Inválida");
}
```

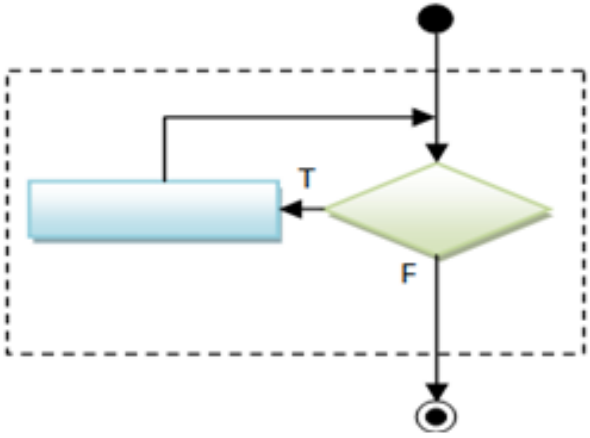
Ejercicios para estructuras selectivas

1. Escribe un programa que lea de teclado un numero entero y determine si es par o impar.
2. Escribe un programa que lea una calificación de un alumno y obtenga como salida un mensaje si fue aprobado o reprobado (70 o mas es aprobado).
3. Escribe un programa que reciba un número del 0 al 10 y regrese como resultado el número en letras.
4. Escribe un programa que lea un numero y si este es par obtenga su mitad, si es impar, obtenga su cuadrado.
5. Escribe un programa que lea un dato numérico y dependiendo del rango donde se encuentre obtenga el resultado:

Rango	resultado
0-5	Malo
6-7	Regular
8-9	Bueno
10	Excelente

6. Escribe un programa que lea de teclado dos números que representan día y mes y determine la estación del año en la que se encuentra esa fecha.
7. Escribe un programa que lea de teclado un número y obtenga el mes que representa.
8. Escribe un programa que lea de teclado un numero entero y si esta entre 0 y 99, obtenga el número de decenas que contiene.
9. Escribe un programa que les 2 datos nombre y edad de 2 personas y obtenga el nombre de la mas pequeña.
10. Escribe un programa que presente un menú de opciones y obtenga el área de la figura, después de leer los valores necesarios:
    - 1 triángulo
    - 2 cuadrado
    - 3 circulo
    - 4 rectángulo
    - 5 salir

3.3 Estructuras iterativas: repetir mientras, hasta, desde



Iterativa

- Repetición fija (for)
- Repetición por elementos (for)
- Repetición condicional de entrada (while)
- Repetición condicional de salida(do- while)

Instrucción For

La instrucción **for** provee una forma compacta de iterar sobre un rango de valores. La forma general de la instrucción **for** puede ser expresada así:

for (*initialization*; *termination*; *increment*)
    statement

Ej. ForDemo.java

```
for (int i=1; i<11; i++)
        System.out.println("Count is: " + i);
```

- Los ciclos **for** son frecuentemente utilizados para iterar sobre los elementos de un arreglo, o los caracteres de un String.
- ***initialization*** es una expresión que inicializa el ciclo. Es ejecutada una vez al comienzo del ciclo.
- ***termination*** determina cuando terminar el ciclo. Esta expresión es evaluada al tope de cada iteración del ciclo. Cuando la expresión evalúa a **falso**, el ciclo termina.
- ***increment*** es invocada después de cada iteración.
- **Todos esos elementos son opcionales.**

```
// infinite loop
for ( ; ; ) {
    // your code goes here
}
```

Ejemplo usando for

```
// imprime los primeros n numeros enteros
for (int i=0; i<=n; i++)
  System.out.println(i);

// imprime los primeros n numeros enteros pares
for (int i=0; i<=n; i++)
  System.out.println((i*2));

// suma los primeros n numeros enteros
for (int i=0; i<=n; i++)
  sum = sum + i;
System.out.println(sum);
```

Repetición por elementos (for)

- La instrucción **for** puede ser utilizada con otro formato que permite **acceder a los elementos de una colección** o de un arreglo de manera directa sin referirse a su posición de almacenamiento.

for (tipo *variable : coleccion*) {

    // estatutos para acceder a los elementos de *coleccion*

}

```
class EnhancedForDemo {
    public static void main(String[] args){
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};
        for (int item : numbers) {
          System.out.println("Count is: " + item);
        }
    }
}
```

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html

Repetición por elementos (for)

```
int i=0;
int [] horas= new int[7];
String [] dias= {"Lunes", "martes", "Miércoles" };
for (String item : dias) {
      System.out.println("Horas trabajadas  el " + item);
      horas[i++] = Keyboard.readInt();
}
```

While y Do-While

Se debe usar una instrucción **while** para continuamente ejecutar un bloque de instrucciones mientras una condición permanezca verdadera. La sintaxis general de while es:

```
while (expression) {
  estatuto
}
```

Primero, la instrucción while evalúa la expresión, la cual debe retornar un valor booleano. Si la expresión retorna verdadero, entonces la instrucción while ejecuta las instrucciones dentro del bloque asociado. El proceso se mantiene ininterrumpido hasta que la expresión retorne falso.

Ej. WhileDemo.java

Ejemplo usando while

```
// imprime los primeros n numeros enteros
int i=0;
while (i<=n) {
  System.out.println(i);
  i++;
}

// ciclo infinito
while (true) {
  System.out.println(i);
  i++;
}

while (!bandera) {
  …
}
```

Do - While

Esta instrucción es muy similar a la anterior, con la diferencia de que la evaluación de la instrucción se hace al final, no al principio. Esto permite ejecutar por lo menos una vez el bloque asociado sin evaluar la expresión.

```
do
   estatuto;
while (expresionBooleana);
```

Ej: DoWhileDemo.java

Ejemplo usando do while

```
// imprime al menos una vez porque la condición se evalúa al final
int i=10;
do {
    System.out.println(i);
     i++;
} while (i<=5) ;
```

```
class DoWhileDemo {
    public static void main(String[] args){
        int count = 1;
        do {
            System.out.println("Count is: " + count);
            count++;
        } while (count <= 11);
    }
}
```

```
// Convertidor de grados
// Luis P.
// FP 2014

import java.util.Scanner;
public class Grados {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double gradosC, gradosF;
        do {
            System.out.println("Introduce grados Centígrados:");
            gradosC = sc.nextDouble();
            gradosF = 32 + (9 * gradosC / 5);
            System.out.println(gradosC +" ºC = " + gradosF + " ºF");
        } while (gradosC != 0);
  }
}
```

Ejercicios para estructuras de repetición

1. Escribe un programa que lea de teclado x números enteros hasta que el número leído sea 0.
2. Escribe un programa que lea de teclado la edad de 10 personas y obtén como resultado el promedio de las edades.
3. Escribe un programa que lea de teclado un dato entero y determine si este es o no un número primo. Los números primos son aquellos que solo tienen dos divisores (ellos mismos y la unidad)
4. Escribe un algoritmo que obtenga el número de letras minúsculas que contiene un String .
5. Escribe un programa que lea de teclado un numero de 0 a 10 y obtenga su factorial.
6. Escribe un programa que lea de teclado un numero e imprima su tabla de multiplicar desde 0 hasta 12.
7. Escribe un programa que lea de teclado un numero de 0 a 10 y obtenga todas las tablas de multiplicar desde 0 hasta el numero leído.

8. Escribe un programa que lea de teclado nombre y sexo de 10 personas y obtén como resultado el numero de mujeres.
9. Escribe un programa que lea de teclado nombre y sexo de 10 personas y obtén como resultado el total de hombres y el total de mujeres.
10. Escribe un programa que lea de teclado nombre, edad y sexo de 10 personas y obtén como resultado el numero de mujeres, el promedio de edad de las mujeres y el nombre de la mas joven.
11. Escribe un programa que lea de teclado 10 nombres de personas y obtenga como resultado cuantos nombres inician con letra A
12. Escribe un programa que lea de teclado un número entero y regrese como resultado en pantalla todos sus divisores.
13. Escribe un programa que lea de teclado un String y regrese un mensaje indicando si el String es un palíndromo.
14. Escribe un programa que lea de teclado un String y obtenga como resultado el número de palabras que contiene

15. Escribe un programa que lea de teclado nombre y edad de 20 personas y obtenga el nombre de la persona mayor.
16. Escribe un programa que lea de teclado x números enteros hasta que el número leído sea 0.
17. Escribe un programa que lea de teclado un String y obtenga el total de caracteres que son dígitos.
18. Escribe un programa que lea de teclado un número entero y obtenga como resultado un valor boolean, verdadero si el número es par y falso si es impar.
19. Escribe un programa que lea de teclado un entero y determine si este es o no un número primo. Los números primos son aquellos que solo tienen dos divisores (ellos mismos y la unidad)
20. Escribe un programa que lea de teclado un número del 0 al 10 y regrese como resultado el número en letras.
21. Escribe un programa que presente un menú de opciones (hasta que la opción sea salir) y obtenga el área de la figura, después de leer los valores necesarios:
    - 1 triángulo
    - 2 cuadrado
    - 3 circulo
    - 4 rectángulo
    - 5 salir

Ejercicio

- La CFE desea obtener los recibos de pago de sus usuarios, la cantidad a pagar se calcula con base a los kw consumidos en el periodo de acuerdo a la siguiente tabla, además de agregar el 16% de iva.

Kw	Costo
0-299	$0.45
300-450	$0.68
451 en adelante	$0.95

- Escribe un programa para procesar los datos de 50 usuarios (nombre, domicilio, numero de medidor, lectura anterior y lectura actual) y obtenga los recibos de pago de los usuarios.

3.4. Diseño e implementación de funciones

- Los programas Java pueden contener métodos para especificar el comportamiento de un objeto.
- Los métodos pueden pertenecer a la clase (static) y estos se usan para implementar funciones que siempre se ejecutan de la misma forma y normalmente usan valores para obtener resultados.
- Ejemplo:
  - Math.sin(x); // **sin** es un método estático de la clase **Math**
- Las clases pueden contener métodos para actualizar o consultar los datos de un objeto:
  - get( ) se utilizan para obtener el dato almacenado
  - set (valor) se utiliza para almacenar un valor

Ejemplo

```
public class Punto {
   protected int x;
   protected int y;
// constructor
  public Punto (int valorX, int valorY) {
        x=valorX;
        y=valorY;
   }
   public void setX(int valorX) {
        x=valorX;
   }
   public int getX() {
        return x;
   }
// devolver la representacion a String del objeto
   public String toString() {
         return "[" +x +", "+y+" ]";
    }
}
```

Ejercicio

- Escribir la clase Punto:
  - Incluir los métodos setY y getY
  - Incluir un atributo llamado color con sus métodos setColor y getColor.
  - Agrega también un constructor que contenga 3 parámetros (valorX, valorY, color)
- Compilar la clase Punto.
- Escribir una clase para usar la clase Punto.
  - En el método main, deberás crear tres objetos de tipo Punto con diferentes valores para x, y y color.
  - Imprimir los tres objetos, cambiar algún valor a los objetos.
  - Imprimir los objetos que son de color “rojo”.

Portafolio

- Temas de la unidad
- 10 programas de selección
- 15 programas de repetición
- 5 clases con métodos (get, set y cálculos)