Programación Interactiva
Fundamentos de Programación

Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle

Condicionales

- Ejecución condicional: **if**

```
if (<condicion>)
{
    <sentencia>;
}
else
{
    <sentencia>;
}
```

- No hay instrucción ***elif*** o ***elsif*** como en otros lenguajes, pero es idéntico a un ***"else if"***

Condicionales

- Ejemplo:

```
if (DEBUG)
   System.out.println("DEBUG: x = " + x);
```

```
if (botónPulsado == botónOK)
{
   // Código para el botón OK
}
else if (botónPulsado == botónCancel)
{
   // Código para el botón Cancel
}
```

Condicionales

- Ejecución condicional múltiple: **switch**

```
switch (<variable>)
{
   case <valor 1>:
                        <sentencia>;
                        break;
   case <valor 2>:
                        <sentencia>;
                        break;
   default:
                        <sentencia>;
}
```

- No es obligatorio colocar el ***default*** ni los ***break***
- ***"variable"*** puede ser de tipo ***int***, ***enum*** o ***char***

Condicionales

- Ejecución condicional múltiple: **switch**

```
switch (estado_matricula)
{
    case 'R':
       return "Repitente";
    case 'C':
       return "Cancelada";
    default:
       return "Primera vez";
}
```

```
if (estado_matricula=='R')
    return "Repitente";
else if (estado_matricula=='C')
    return "Cancelada";
else
    return "Primera vez";
```

- No es obligatorio colocar el ***default*** ni los ***break***
- ***"variable"*** puede ser de tipo ***int***, ***enum*** o ***char***

Ciclos

- **for**
  - Se hasta que la condición sea verdadera, además, puede ejecutar una instrucción adicional por cada iteración:

```
for (<inicialización>; <condición>; <instrucción>)
{
        <sentencias>;
}
```

```
int numero = 6, factorial = 1;

for (int i=2; i<=numero; i++)
{
   factorial *= i;
}

System.out.println("El factorial de "+numero+" es"
   +factorial);
```

Ciclos

- **for, foreach, sintaxis alternativa**
  - Hay una sintaxis alternativa para el ***for*** cuando se desea iterar sobre los elementos de un ***array*** como en el caso anterior:

```
// Devuelvo "true" si "num" se encuentra en el array "arrayNums",
// "false" en caso contrario
int isInArray (int num, int[] arrayNums)
{
   for (int i : arrayNums)
      if (i==num)
         return true;
   return false;
}
```

Esta sintaxis solo está disponible desde la versión 1.5.0 de Java

Ciclos

- **while**
  - Se repite mientras la condición sea verdadera

```
while (<condición>)
{
    <sentencias>;
}
```

```
int numero = 6, factorial = 1;

while (numero!=1)
{
    factorial *= numero;
    numero--;
}

System.out.println("El factorial de "+numero+" es"+
    factorial);
```

Ciclos

- **while**
  - Ejemplo práctico:

```
FileReader in = new FileReader(new File("archivo.txt"));
int c;
while ((c = in.read()) != -1)
    System.out.print(c);
in.close();
```

Imprimir el contenido de un archivo en la consola

Ciclos

- **do … while**
  - Igual que el ***while***, pero la condición se evalúa al final de cada iteración:

```
int calificación = 0;
while (calificación!=0)
{
    System.out.println("Esto nunca se imprime");
}
```

  - Mientras que con un ***do…while***

```
int calificación = 0;
do {
    System.out.println("Se imprime una vez");
} while (calificación!=0);
```

continue

- **continue**
  - Termina la iteración actual en un ciclo, comenzando la iteración siguiente.

```
// Solo imprimo los impares múltiplos de 3
for (int impar=1; impar<=19; impar+=2)
{
   if (impar%3 != 0)
      continue;

   System.out.println("El "+impar+" es divisible por 3");
}
```

  - Imprimiría:

El 3 es divisible por 3
El 9 es divisible por 3
El 15 es divisible por 3

break

- **break**
  - Termina la ejecución de un ciclo, saliendo totalmente de él. También funciona en los ***switch***.

```
// Solo imprimo el primer múltiplo de 3
for (int impar=1; impar<=19; impar+=2)
{
   if (impar%3 == 0)
   {
      System.out.println("El "+impar+" es divisible por 3");
      break;
   }
}
```

  - Imprimiría:

    El 3 es divisible por 3

return

- **return**
  - Termina la ejecución de un método, saliendo de él hacia el método que lo invocó. Mediante el return se puede retornar opcionalmente un valor dependiendo del tipo del método donde es invocado.

```
// Obtengo la posición de "num" en el array "arrayNums"
int getPosicion (int num, int[] arrayNums)
{
   for (int i=0; i!=num.size; i++)
      if (arrayNums[i]==num)
         return i;
   return -1;
}
```

arrays

- Un *array* es una colección ordenada de valores de cualquier tipo

- Por ejemplo, un array que tenga espacio para 10 números enteros se declara así:

```
int miArray[] = new int[10];
```

- Para acceder a cada elemento del ***array*** se debe usar el nombre del ***array*** seguido de corchetes y el índice del elemento.

- Los índices de los elementos comienzan en 0. Es decir que el primer elemento del anterior array es “miArray[0]”, y no “miArray[1]” como podría pensarse

arrays

- Un ***array*** es un objeto, como pueden darse cuenta por el uso del operador ***new***

- Para inicializar los elementos de un ***array*** se pueden colocar los valores encerrados entre llaves después de ***new***:

```
int miArray[] = new int[] {1,3,5,7,9};
```

- O tan solo colocando los elementos dentro de llaves así:

```
int miArray[] = {1,3,5,7,9};
```

Paso de arrays a métodos

- Un método que recibe un array puede declararse dos maneras:
  - ***void*** metodoA (***double*** numeros[]) {}
  - ***void*** metodoA (***double***… numeros) {}

- La segunda sintaxis funciona igual que la primera, con la adición de que los elementos del array “numeros” pueden ser pasdos uno por uno en la llamada a “metodoB”:
  - MetodoB (1,2,3,4,5);

Enumeraciones: *enum*

- Una enumeración es un conjunto de valores con nombre.
- Las enumeraciones en Java fueron agregadas en la versión 1.5.0 del lenguaje
- Por ejemplo, si quisiéramos guardar los posibles estados de una cuenta de ahorros podríamos crear una enumeración así:

```
enum ESTADOS_CUENTA { ACTIVA,BLOQUEADA,CANCELADA,EMBARGADA };
```

Enumeraciones: *enum*

- Luego, la enumeración puede usarse en un ***switch***, un ***while***, ser asignada a una variable, o con cualquier otra secuencia de control así:

```
enum ESTADOS_CUENTA { ACTIVA,BLOQUEADA,CANCELADA,EMBARGADA };

ESTADOS_CUENTA estadoCuenta = ESTADOS_CUENTA.ACTIVA;

switch (estadoCuenta) {
   case ACTIVA:
      System.out.println("Si, podemos darle dinero");
   default:
      System.out.println("La transacción no se puede continuar");
}
```

Métodos

- Los métodos de un objeto son las funciones que este puede ejecutar.

- Los métodos *constructores* son métodos que se ejecutan inmediatamente se crea un objeto. Estos métodos “construyen” o adecúan el estado interno del objeto (creación de una interfaz, inicialización de la configuración de un objeto desde un archivo, etc).

- Los constructores deben tener el mismo nombre de la clase:

```
public nombreClase (tipo nombreArg1, tipo nombreArg2 ...){
}
```

Métodos

- Los métodos no constructores deben declararse con un tipo de retorno, a diferencia de los contructores:

```
public tipo nombreClase (tipo nombreArg1, tipo nombreArg2 ...){
}
```

- Por ejemplo:

```
public int calcularArea(){
}
public Vehiculo (int d, int t){
}
```

Método Main

- El método especial *main* es el punto de entrada de un programa en Java, es decir, es el primer método que se ejecuta cuando se invoca a la JVM:

```
[usuario]# java ClasePrincipal
```

- La síntaxis de este método puede ser una de estas dos:

```
public static void main (String[ ] args){
}
public static void main (String… args){
}
```

Ejercicio

- Realizar un programa en Java que permita calcular el mayor y el menor de dos números enteros

3. Entender el Problema: mostrar el mayor y el menor de dos números enteros, para lo cual el usuario debe digitar un par de números, y nosotros debemos mostrar el resultado de la función en pantalla.

    - Para capturar información del usuario:

      JOptionPane.showInputDialog();

    - Para mostrar información en al pantalla:

      JOptionPane.showMessageDialog();

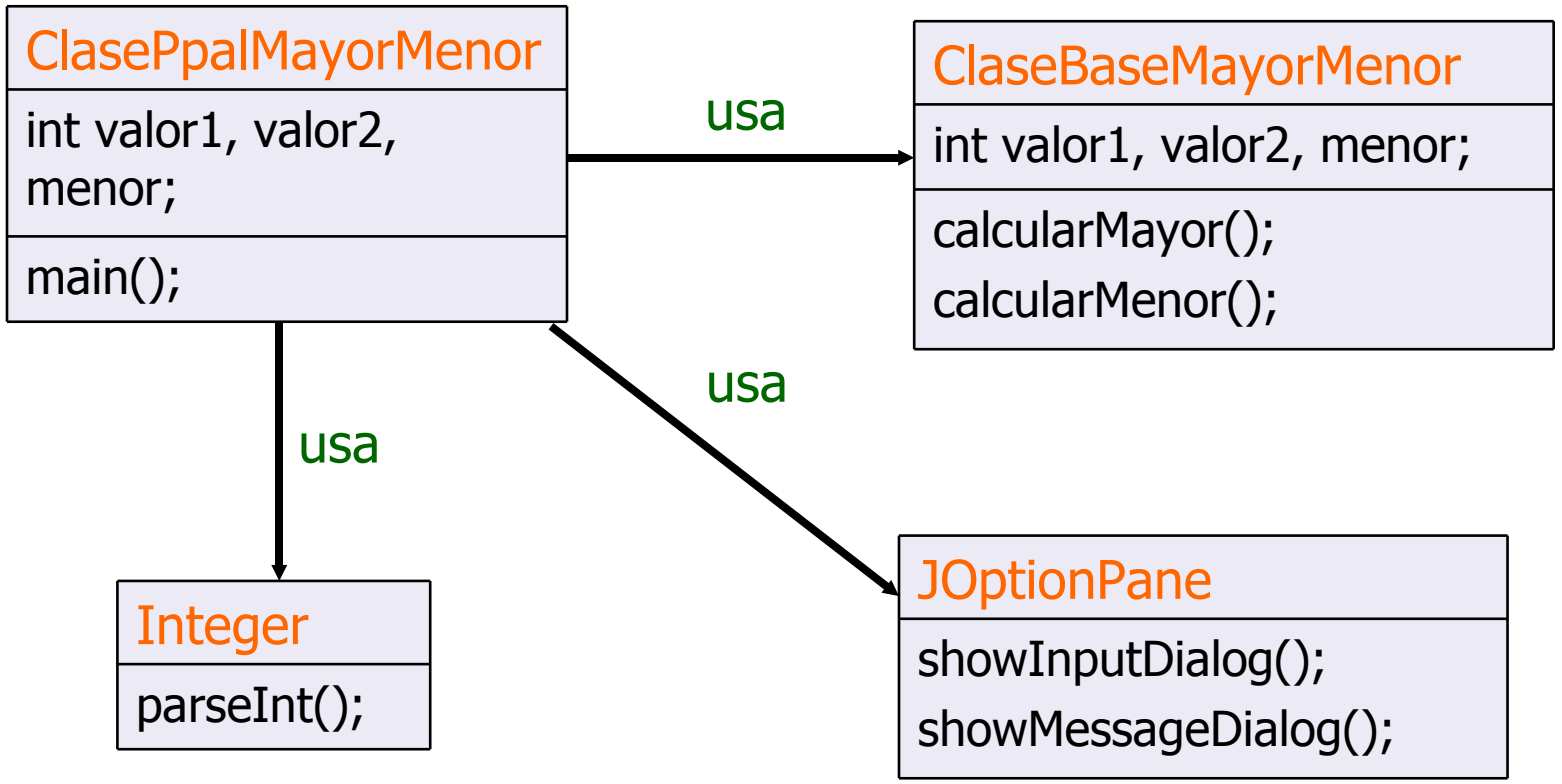
Ejercicio

1. Definir los objetos del dominio del Problema:

   - ClaseBaseMayorMenor
   - ClasePrincipalMayorMenor
   - Integer
   - JOptionPane

Ejercicio

1. Crear el modelo de Objetos :



Ejercicio

1. Definir Algoritmo dentro de los métodos:

- Clase ClaseBaseMayorMenor, método calcularMayor:
  - comparar valor1 y valor2
  - si valor1 es mayor que valor2 entonces valor1 es el mayor
  - si valor2 es mayor que valor1 entonces valor 2 es el mayor
  - si los dos son iguales entonces no hay mayor, devolvemos cualquiera

Ejercicio

1. Definir Algoritmo dentro del main:

- Clase ClasePpalMayorMenor, método main:
  - **Leer** valor1 y valor2
  - **Pasar** valor1 y valor2 a la clase ClaseBaseMayorMenor
  - **Ordenar** a ClaseBaseMayorMenor que calcule el mayor
  - **Ordenar** a ClaseBaseMayorMenor que calcule el menor
  - **Imprimir** el mayor y el menor

Ejercicio 2

- Realizar un programa tipo aplicación que permita calcular el promedio de la nota de un estudiante del curso de algoritmia, a parir de tres notas: nota1, nota2, nota3. Si el estudiante obtuvo entre 0 y 2.9 se debe imprimir que el estudiante obtuvo una D (deficiente), si obtuvo entre 3.0 y 3.9 se imprime que tuvo una B (bueno) y si la nota fue de 4.0 a 5.0 entonces obtuvo una E (excelente).

Ejercicio 3

- Realizar una aplicación en Java que permita encontrar el mayor de n números enteros ingresados por el usuario Utilizar la instrucción de repetición for.

- Tener en cuenta estas 2 situaciones:

  - Al inicio de la ejecución del programa este debe preguntarle al usuario cuantos números va a ingresar
  - El usuario no dice cuantos números va a ingresar, solo los ingresa hasta que dé click en el botón Cancelar en el JOptionPane.

Tarea

Tener claros los siguientes conceptos:

- Herencia
- Polimorfismo
- Encapsulación

Tarea

Entender qué sucedería al ejecutar el siguiente fragmento de código:

```
public class Circulo {
	double radio;
	public Circulo(double r) {
		radio = r;
	}
	public double getArea() {
		return Math.PI*radio*radio;
	}
	public static void main(String[] args) {
		Circulo c = new Circulo(4);
		System.out.println(c.getArea());
	}
}
```

“Tarea”

- Leer sobre la sintaxis básica de Java:
  - declaración de variables y funciones
  - modificadores de visibilidad: *public*, *protected*, *private*
  - para qué sirve el *main* ?

- **Próxima clase: QUIZ**
  - sobre los conceptos vistos en estas 2 clases