

Módulo 6 Strings y Condiciones

¿Qué es un String?

String

Un String es un conjunto de caracteres. No es una variable primitiva, pues está compuesta de por lo menos dos caracteres.

Veamos el siguiente ejemplo:

```
public static void main(String[] args) {
    String game = "Super Mario World";
}
```

```
public static void main(String[] args) {
    String game = "Super Mario World";
DATA
                                                            W
                                                  10 11 12 13
                                                                   14 | 15
INDEX
                          4
                                           8
```



Importante dos cosas:

- El conteo de los índices comienza en 0. El caracter 'S' es el caracter en el índice 0.
- Para indicar que el String termina, Java almacena el caracter NULL (\0) al final de la cadena.

Un String se puede declarar de tres formas distintas:

```
01  String g1;
02  g1 = "Hello!";
03
04  String g2 = "Hello!";
05
06  String g3 = new String("Hello!");
```

Las variables de tipo String pueden almacenar cualquier caracter de Unicode utilizando representación UTF-16 [16 bits].

Concatenar Strings

Dos strings pueden concatenarse utilizando el operador +.

Podemos concatenar valores y variables de diferentes tipos, como Strings, int, double, char, boolean, etc.

```
01 String greeting;
02 greeting = "Hello ";
03
04 String sentence;
05 sentence = greeting + "officer";
06 System.out.println(sentence); //Prints "Hello officer"
```

Inicializar un String

Hay veces que es necesario incializar un String con un valor vacío. Esto se puede hacer de la siguiente forma:

```
01 String s1 = "";
```

String length()

El método **length**() nos permite calcular el tamaño de un String. Este método devolverá un entero con la cantidad de caracteres que un String contenga.

```
01 String s1;
02 int n;
03
04 	 s1 = "12345";
05 n = s1.length();
06 System.out.println(n); //Prints 5
07
08 	 s1 = "aabc12345";
09 n = s1.length();
10 System.out.println(n); //Prints 9
11
12 	 s1 = "";
13 n = s1.length();
14 System.out.println(n); //Prints 0
```

String substring()

El método substring (int beginIndex, int endIndex) regresa un recorte del String original. El substring comienza en el índice especificado como beginIndex y se extiende hasta endIndex – 1. El tamaño del String resultante será: endIndex – beginIndex.

NOTA: El conteo de posiciones comienza desde 0.

Ejemplo:

```
01 String s1 = "Monterrey, Nuevo León";
02 String ciudad = s1.substring(0,9);
03 String estado = s1.substring(11,21);
04
05 System.out.println(ciudad); //Prints "Monterrey"
06 System.out.println(estado); //Prints "Nuevo León"
07
08 String s2 = "smiles".substring(1, 5);
09 System.out.println(s2); //prints "mile"
```

String charAt()

El método charAt(int index) permite recuperar el caracter que se encuentra en la posición especificada, siendo 0 el índice del primer caracter, y length()-1 índice del último.

```
01 String s1 = "The Jungle Book";
02 char c1 = s1.charAt(1);
03 System.out.println(c1); //prints 'h'
04
05 char c2 = s1.charAt(4);
06 System.out.println(c2); //prints 'J'
07
08 char c3 = s1.charAt(s1.length()-1);
09 System.out.println(c3); //prints 'k'
```

String trim()

El método trim() crea un nuevo string eliminado los espacios vacíos al inicio y al final.

```
01 String s1 = " :) ";
02 String s2 = s1.trim();
03 System.out.println(s2); //prints ":)"
```

String toLowerCase() / toUpperCase()

Los métodos toLowerCase() y toUpperCase() nos permiten procesar un texto para convertir una cadena a minúscula o mayúscula respectivamente. Estos métodos generan cadenas nuevas, por lo que debemos asignar el valor resultante a una variable de tipo String.

```
01 String s1 = "abCD";
02 String s2 = "abCD";
03
04 String lowerCase = s1.toLowerCase();
05 String upperCase = s2.toUpperCase();
06
07 System.out.println(lowerCase); //Prints "abcd"
08 System.out.println(upperCase); //Prints "ABCD"
```

String replace()

El método replace (char oldChar, char newChar) sirve para reemplazar un caracter dentro del texto especificado por otro.

```
01 String s1 = "Bienvenido a la ciudad!";
02 String s2 = s1.replace('e','x');
03 System.out.println(s2); //Prints "Bixnvxnido a la ciudad!"
04
05 s2 = s2.replace('a','x');
06 System.out.println(s2); //Prints "Bixnvxnido x lx ciudxd!"
```

String trim()

El método trim() crea un nuevo string eliminado los espacios vacíos al inicio y al final.

```
01 String s1 = " :) ";
02 String s2 = s1.trim();
03 System.out.println(s2); //prints ":)"
```

Condiciones

$_{ m IF}$

Existen instrucciones de control de flujo que nos permiten ejecutar selectivamente ciertos bloques de código. La primera instrucción es el if.

```
01 int result;
02 if (denominator != 0) {
03   result = numerator / denominator;
04 }
```

Cuando la condición que está dentro de los paréntesis se cumple, entonces las instrucciones dentro del bloque { } se ejecutarán.

IF-ELSE

Podemos incluir un segundo bloque, el ELSE, que se ejecuta cuando la condición NO se cumple.

Un bloque ELSE siempre debe estar ligado a un bloque IF.

```
01 int result;
02
03 if (denominator == 0) {
04    System.out.println("Error! You cannot divide by zero.");
05 } else {
06    result = numerator / denominator;
07 }
```

Expresiones boleanas

Dentro de un IF, debemos poner una expresión boleana. Estas pueden ser expresiones boleanas simples:

```
if (minutes > 60){
    hour++;
}

if (letter == 'c') {
    System.out.println("You selected option c");
}
```

O expresiones boleanas complejas:

```
if ((hour >= 8) && (hour <= 22)){
    System.out.println("The store is open!");
}

if (letter == 'a' || letter == 'e' || letter == 'i' || letter == 'o' || letter == 'u' ){
    System.out.println("Letter is a vocal!");
}</pre>
```

| Name | Java Notation | Java Examples |
|-------------|---------------|------------------------------------|
| Logical and | && | (sum > min) && (sum < max) |
| Logical or | 11 | (answer == 'y') (answer == 'Y') |
| Logical not | ! | !(number < 0) |

Operadores boleanos

| Math Notation | Name | Java Notation | Java Examples |
|------------------|--------------------------|------------------|--------------------------------|
| = | Equal to | == | balance == 0 answer == 'y' |
| ≠ | Not equal to | != | income != tax answer != 'y' |
| > | Greater than | > | expenses > income |
| ≥ | Greater than or equal to | >= | points >= 60 |
| < | Less than | < | pressure < max |
| ≤ | Less than or equal to | <= | expenses <= income |

| Value of A | Value of B | Value of A && B | Value of A B | Value of ! (A) |
|------------|------------|-----------------|---------------------|----------------|
| true | true | true | true | false |
| true | false | false | true | false |
| false | true | false | true | true |
| false | false | false | false | true |

Comparación de variables primitivas

Para comparar la igualdad de dos valores primitivos, podemos utilizar el operador de igualdad (==).

Por ejemplo:

```
01 if (a == 3) {
02    System.out.println("a equals 3");
03 }
```

Comparación de Strings

Para comparar dos variables de tipo String, NO podemos utilizar el operador (==), pues no son variables primitivas.

Para esto, debemos utilizar el método equals().

```
01 String s1 = "hola";
02 String s2 = "adios";
03
04 if (s1.equals(s2)){
05    System.out.println("Iguales!");
06 } else {
07    System.out.println("Diferentes!");
08 }
```

Comparación de Strings

También podemos utilizar el método equalsIgnoreCase() si queremos verificar comparar sin considerar mayúsculas o minúsculas.

```
01 String s1 = "hola";
02 String s2 = "HOLA";
03
04 if (s1.equalsIgnoreCase(s2)){
05    System.out.println("Iguales!");
06 } else {
07    System.out.println("Diferentes!");
08 }
```

Ejercicio!

Ejemplo

Monkey Trouble!

Tenemos dos changos:

- Si los dos changos están sonriendo, estamos en problemas
- Si los dos changos están serios, estamos en problemas.

Diseña un programa de Java que permita modelar este ejercicio.



```
01 import java.util.Scanner;
02
   public class MonkeyTrouble {
       public static void main(String[] args) {
04
           Scanner teclado = new Scanner(System.in);
05
06
           System.out.print("Is the first monkey smiling? (true/false): ");
07
           boolean monkey1 = teclado.nextBoolean();
08
09
           System.out.print("Is the second monkey smiling? (true/false): ");
10
           boolean monkey2 = teclado.nextBoolean();
11
12
13
           if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
               System.out.println("Look out! The monkeys are planning something!");
14
15
           }
16
           else {
               System.out.println("Don't worry, everything is OK");
17
18
19
           teclado.close();
20
21 }
```

```
01 import java.util.Scanner;
02
   public class MonkeyTrouble {
        public static void main(String[] args) {
04
20
21 }
```

```
Scanner teclado = new Scanner(System.in);
05
06
           System.out.print("Is the first monkey smiling? (true/false): ");
07
            boolean monkey1 = teclado.nextBoolean();
08
09
           System.out.print("Is the second monkey smiling? (true/false): ");
10
            boolean monkey2 = teclado.nextBoolean();
11
12
```

Preguntar y capturar el estado de los dos monos. Al ser variables boolean, hay que capturar TRUE / FALSE

```
if ((monkey1 == true && monkey2 == true) || (monkey1 == false && monkey2 == false)) {
13
               System.out.println("Look out! The monkeys are planning something!");
14
15
```

Si ambos monos, o ninguno, están sonriendo, entonces imprimimos en pantalla un mensaje de alerta.

```
else {
16
                System.out.println("Don't worry, everything is OK");
17
18
```

De lo contrario, imprimimos que todo está OK

```
teclado.close();
19
```

Como la variable monkey es una variable de tipo boolean que ya almacena un valor TRUE/FALSE, podemos ahorrarnos la comparación.

Y utilizar la siguiente expresión:

```
01 if ((monkey1 && monkey2) || (!monkey1 && !monkey2)) {
02    System.out.println("Look out! The monkeys are planning something!");
03  }
04  else {
05    System.out.println("Don't worry, everything is OK");
06 }
```

Ambas expresiones son equivalentes.

Switch

| Lunes | Martes | Miércoles y Jueves | Viernes | Sábado y Domingo |
|--|---------|-----------------------|---------|---------------------|
| 2X1 OUNT OF THE PARTY OF THE P | | ENEWA TERES | | |
| | \$49.00 | \$35.00 | \$30.00 | No hay promociones! |
| | | | | |
| | | | | |

Usando IFs

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: ");
String day = keyboard.nextLine();
if (day.equals("Monday")){
  System.out.println("2x1 in movie tickets!");
if (day.equals("Tuesday")){
  System.out.println("Popcorn for $49.00");
if (day.equals("Wednesday") || day.equals("Thursday")){
  System.out.println("Movie tickets for $35.00");
if (day.equals("Friday"))
  System.out.println("Any coffee for $30.00");
if (day.equals("Saturday") || day.equals("Sunday")){
  System.out.println("No promos!");
keyboard.close();
```

Usando IFs

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: ");
String day = keyboard.nextLine();
if (day.equals("Monday")){
  System.out.println("2x1 in movie tickets!");
if (day.equals("Tuesday")){
  System.out.println("Popcorn for $49.00");
if (day.equals("Wednesday") || day.equals("Thursday")){
  System.out.println("Movie tickets for $35.00");
if (day.equals("Friday"))
  System.out.println("Any coffee for $30.00");
if (day.equals("Saturday") || day.equals("Sunday")){
  System.out.println("No promos!");
keyboard.close();
```

Problemas:

- 1. El programa no detecta escenarios en donde los días están mal escritos.
- 2. Debemos agregar una condición por cada caso que debamos considerar.
- 3. Largo y difícil de leer.

Usando IF-ELSE

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: ");
String day = keyboard.nextLine();
if (day.equals("Monday")){
  System.out.println("2x1 in movie tickets");
} else if (day.equals("Tuesday")){
  System.out.println("Popcorn for $49.00");
} else if (day.equals("Wednesday") || day.equals("Thursday")){
  System.out.println("Movie tickets for $35.00");
} else if (day.equals("Friday")){
  System.out.println("Any coffee for $30.00");
} else if (day.equals("Saturday") || day.equals("Sunday")){
  System.out.println("No promos :(");
} else {
  System.out.println("Invalid day! try again");
keyboard.close();
```

Usando IF-ELSE

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: ");
String day = keyboard.nextLine();
if (day.equals("Monday")){
 System.out.println("2x1 in movie tickets");
} else if (day.equals("Tuesday")){
 System.out.println("Popcorn for $49.00");
} else if (day.equals("Wednesday") || day.equals("Thursday")){
 System.out.println("Movie tickets for $35.00");
} else if (day.equals("Friday")){
 System.out.println("Any coffee for $30.00");
} else if (day.equals("Saturday") || day.equals("Sunday")){
 System.out.println("No promos :(");
} else {
 System.out.println("Invalid day! try again");
keyboard.close();
```

Problemas:

- Debemos agregar una condición por cada caso que debamos considerar.
- 2. Largo y difícil de leer.

Switch

Un switch es una instrucción condicional que nos permite determinar múltiples caminos a partir de una expresión integral.

Podemos utilizar una variable numérica o texto para generar los distintos caminos.

El switch está compuesto por tres elementos:

- 1. Switch
- 2. Case
- 3. Default (opcional)

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: "):
String day = keyboard.nextLine()
switch(day){
 case "Monday":
   System.out.println("2x1 in movie tickets");
   break;
 case "Tuesday":
   System.out.println("Popcorn for $49.00");
   break;
 case "Wednesday":
   System.out.println("Movie tickets for $35.00");
   break;
 case "Friday":
   System.out.println("Any coffee for $30.00");
   break;
keyboard.close();
```

Especificamos la condición usando: switch(<expresion integral>)

Especificamos cada camino utilizando la sintaxis:

case <caso>:

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: ");
String day = keyboard.nextLine();
switch(day){
 case "Monday":
   System.out.println("2x1 in movie tickets");
   break;
 case "Tuesday":
   System.out.println("Popcorn for $49.00");
   break;
 case "Wednesday":
    System.out.println("Movie tickets for $35.00");
   break;
 case "Friday":
   System.out.println("Any coffee for $30.00");
   break;
keyboard.close();
```

La sentencia break se agrega para salir del bloque switch {}

Si no agregamos esta instrucción, el programa se saltaría a los siguientes case

```
Scanner keyboard = new Scanner(System.in);
System.out.print("What day is today?: ");
String day = keyboard.nextLine();
switch(day){
  case "Monday":
   System.out.println("2x1 in movie tickets");
   break;
  case "Tuesday":
   System.out.println("Popcorn for $49.00");
   break:
  case "Wednesday":
  case "Thursday":
   System.out.println("Movie tickets for $35.00")
   break;
  case "Friday":
   System.out.println("Any coffee for $30.00");
   break;
  case "Saturday":
  case "Sunday":
    System.out.println("No promos :(");
   break;
 default:
    System.out.println("Invalid day! try again");
keyboard.close();
```

```
Si queremos especificar múltiples casos con la misma lógica, podemos indicar case <scenario1>: case <scenario2>: case <scenario3>: code; break;
```

También podemos especificar un caso por default:

default:

Errores comunes

- 1. Olvidar el break: Cuando omitimos los breaks, todas las siguientes opciones se ejecutan.
- 2. Repetir opciones: dos cases distintos NO PUEDEN TENER un mismo valor.

```
char traffic_light = 'R';
02
   switch(traffic_light) {
03
     case 'R': //red light
04
       System.out.println("Stop!");
05
     case 'Y': //yellow light
06
       System.out.println("Slow down!");
07
     case 'G': //green light
80
        System.out.println("Go!");
09
10
     default:
       System.out.println("Wrong color!");
11
12 }
```

```
Esto imprimiría:> Stop!> Slow down!> Go!> Wrong color!
```