0.3 Java

String y Random

La clase String

- La clase **`String`** provista por Java brinda facilidades para almacenar y procesar cadenas de caracteres.
- El **estado interno** de una instancia de tipo **`String`** es una secuencia de caracteres.
- Los objetos de tipo **`String`** son inmutables, quiere decir que cuando se opera con **`String`** el resultado es siempre un nuevo objeto de esta clase.
- Una variable de tipo **`String`** referencia a un objeto de este tipo.

La clase String

Declaración, creación e inicialización

```
String cad = "Buenas Buenas ...";

String var1;

var1 = new String("Buenas Buenas ..." );
```

Ahora la variable cad puede recibir mensajes que se ligarán a métodos provistos por la clase String. Ninguno de estos métodos modifica el estado interno de la variable.

La clase String

`length(): entero`	retorna la cantidad de caracteres de una cadena.
`toLowerCase(): String`	retorna la misma cadena pero con todos los caracteres en minúscula.
`toUpperCase(): String`	retorna la misma cadena pero con todos los caracteres en mayúscula.
`trim(): String`	retorna la misma cadena pero sin los espacios del principio y del final.
`charAt(pos:entero): caracter`	retorna el caracter que está en la posición que corresponde al argumento.

La clase String

substring(ini: entero): String	retorna la subcadena a partir de la posición **ini**.
substring(ini:entero, fin:entero): String	retorna la subcadena a partir de la posición **ini** hasta la anterior a la posición **fin**.
indexOf(A: String): entero	retorna la posición de la primera aparición de la subcadena **A** en la cadena.
lastIndexOf(A: String): entero	retorna la posición de la última aparición de la subcadena **A** en la cadena.
compareTo(A: String): entero	Retorna `0` si las cadenas son iguales , o un número positivo o un número negativo según el orden alfabético (minúsculas < mayúsculas).

La clase String

- En Java el mínimo valor para un índice es 0 y corresponde al primer carácter de la cadena.
- Los métodos **`indexOf`** y **`lastIndexOf`** retornan –1 si la subcadena no aparece en la cadena.
- La comparación entre variables de tipo **`String`** no se realiza a través del operador relacional **`==`**, sino con los métodos **`equals(A:String)`** o **`compareTo(A:String)`**.

La clase String

- En Java el mínimo valor para un índice es 0 y corresponde al primer carácter de la cadena.
- Los métodos **`indexOf`** y **`lastIndexOf`** retornan –1 si la subcadena no aparece en la cadena.
- La comparación entre variables de tipo **`String`** no se realiza a través del operador relacional **`==`**, sino con los métodos **`equals(A:String)`** o **`compareTo(A:String)`**.

```
String s1 = new String("Hello world");
String s2 = new String("Hello world");
System.out.println(s1==s2);    FALSE
```

La clase String

Ejemplos

```
String cad = “Buenas Buenas...”;
```

```
cad.length()
retorna 16
```

```
cad.toLowerCase()
retorna “buenas buenas…”
```

```
cad.toUpperCase()
retorna “BUENAS BUENAS…”
```

```
“   Buenas buenas...     ”.trim()
retorna “Buenas buenas…”
```

```
cad.charAt(1)
retorna ‘u’
```

```
cad.charAt(100)
StringIndexOutOfBoundsException: String index out of range: 100 (in java.lang.String)
```

La clase String

Ejemplos

```
String cad = “Buenas Buenas...”;
```

```
cad.substring(3)
retorna “nas Buenas...”
```

```
cad.substring(3,5)
retorna “na”
```

```
cad.substring(3,25)
StringIndexOutOfBoundsException: String index out of range: 25 (in java.lang.String)
```

La clase String

Ejemplos

```
String cad = “Buenas Buenas...”;
```

`cad.indexOf(“Bue”)` `retorna 0`
`cad.indexOf(“Nue”)` `retorna -1`
`cad.lastIndexOf(“Bue”)` `retorna 7`
`cad.compareTo(“Buenos Aires”)` `retorna -14`

La clase String

Mostrar

```
System.out.println (cad) ;
```

Concatenar

```
System.out.println ("El ganador es "+nombre);
```

Conversión implícita y Concatenación

```
System.out.println ("El puntaje es "+10);

int i =0;
System.out.println ("El puntaje es "+i);
```

La clase String

Conversión explícita

Para convertir un número en una cadena de caracteres se emplea el método **valueOf.**

```
int valor=10;
String str= String.valueOf(valor);
```

Para convertir una cadena en un número entero, primero quitamos los espacios en blanco al principio y al final y usamos el método parseInt de la clase Integer

```
String str=" 12 ";
int numero=Integer.parseInt(str.trim());
```

La clase String

Conversión explícita

Para convertir un número en una cadena de caracteres se emplea el método **valueOf.**

```
int valor=10;
String str= String.valueOf(valor);
```

Notar que **valueOf** y **parseInt** son métodos estáticos (**static**) porque son métodos que le pedimos a la clase y no a un objeto de la clase.

Para convertir una cadena en un número entero, primero quitamos los espacios en blanco al principio y al final y usamos el método parseInt de la clase Integer

```
String str=" 12 ";
int numero=Integer.parseInt(str.trim());
```

La clase String

Conversión explícita

Para convertir un string en número decimal se requieren dos pasos:convertir la cadena en un objeto de la clase **`Double`**, mediante el método **`valueOf`**, y a continuación convertir el objeto de la clase **`Double`** en un **tipo primitivo `double`** mediante el método **`doubleValue.`**

```
String str="  12.35 ";
double num=Double.valueOf(str).doubleValue();
```

La clase String

Conversión explícita

Para convertir un string en número decimal se requieren dos pasos:convertir la cadena en un objeto de la clase **`Double`**, mediante el método **`valueOf`**, y a continuación convertir el objeto de la clase **`Double`** en un **tipo primitivo `double`** mediante el método **`doubleValue.`**

```
String str="  12.35 ";
double num=Double.valueOf(str).doubleValue();
```

Notar que **`valueOf`** de la clase **`Double`** es un método estático (**`static`**).

La clase Random

La clase Random

Un **generador** de números aleatorios se utiliza cuando se desea simular **situaciones de azar**.

La clase **`Random`** de Java es un generador de números **pseudo-aleatorios**.

Los números no son realmente aleatorios porque se obtienen a través de un algoritmo que genera una secuencia distribuida uniformemente, a partir de una **semilla** inicial.

La clase Random

La clase brinda dos constructores para crear objetos **`Random`**, uno sin parámetros y otro con un parámetro que establece el valor de la semilla.

Si no se especifica parámetro, el constructor usa la hora actual del sistema como semilla, lo que disminuye la posibilidad de obtener secuencias de números repetidas.

Cuando se crea un objeto **`Random`** con una semilla como parámetro, se puede obtener a continuación lo que parece una secuencia aleatoria, pero si se vuelve a inicializar el objeto con la misma semilla se vuelve a obtener **la misma secuencia**.

La clase Random

1. Importar el paquete que incluye a la clase Random.

```
import java.util.Random;
```

2. Crear un objeto de la clase Random

```
Random rnd = new Random();
Random rnd = new Random(100);
```

3. Invocar uno de los métodos que generan un número aleatorio

```
rnd.nextInt();
rnd.nextInt(3);
rnd.nextFloat();
```

La clase Random

Ejemplo I

Para generar una secuencia de 10 números aleatorios entre 0.0 y 1.0 escribimos

```
for (int i = 0; i < 10; i++) {
   System.out.println(rnd.nextDouble());
}
```

La clase Random

Ejemplo II

Un jugador apuesta una cantidad de dinero y tira una moneda.

Si sale cara obtiene el doble de la cantidad apostada, pero si sale cruz pierde la mitad.

Implemente una simulación para el juego que parta de un valor inicial y lo actualice según en la moneda se obtenga cara o cruz, hasta que llegue a tener $1 o se realicen 50 tiradas.

La clase Random

Ejemplo II

```
import java.util.Random;
import IPOO.ES;
public class DOBLEoMITAD {
    public static void main (String arg[]) {
        Random gen;
        gen = new Random();
        System.out.print("Ingrese la apuesta ");
        int m = ES.leerEntero();
        int i = 0;
        int caracruz;
        …
   }}
```

La clase Random

Ejemplo II

```
while (i<50 && m > 1){
    i++; caracruz = gen.nextInt(2);
    if (caracruz==1){ //cara
        m = m*2;
        System.out.println (i+" cara " +m);
    }
    else{ //cruz
        m = m / 2;
        System.out.println (i+" cruz " +m);
    }
}
```

Demo