SCJP – Capitulo 2

Fundamentos del Lenguaje

Agenda

- Palabras Reservadas
- Rangos y tipos de datos primitivos
- Arreglos, Construcción e Inicialización
- Argunentos de Línea de Comando al método main

Palabras Reservadas

abastract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
final	finally	float	for	goto
if	implements	import	instanceof	int
interface	long	native	new	package
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while	assert	

Modificadores de Acceso

abastract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
final	finally	float	for	goto
if	implements	import	instanceof	int
interface	long	native	new	package
private	**protected**	**public**	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while	assert	

Clases, Métodos y variables

abastract	boolean	break	byte	case
catch	char	**class**	const	continue
default	do	double	else	**extends**
final	finally	float	for	goto
if	**implements**	import	instanceof	int
interface	long	**native**	**new**	package
private	protected	public	return	short
static	**strictfp**	**super**	switch	**synchronized**
this	throw	throws	**transient**	try
void	**volatile**	while	assert	

Control de Flujo

abastract	boolean	**break**	byte	**case**
catch	char	class	const	**continue**
default	**do**	double	**else**	extends
final	finally	float	**for**	goto
if	implements	import	**instanceof**	int
interface	long	native	new	package
private	protected	public	**return**	short
static	strictfp	super	**switch**	synchronized
this	throw	throws	transient	try
void	volatile	**while**	assert	

Captura de Excepciones

abastract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
final	**finally**	float	for	goto
if	implements	import	instanceof	int
interface	long	native	new	package
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	**throw**	**throws**	transient	**try**
void	volatile	while	assert	

Paquetes

abastract	boolean	break	byte	case
catch	char	class	const	continue
default	do	double	else	extends
final	finally	float	for	goto
if	implements	**import**	instanceof	int
interface	long	native	new	**package**
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while	assert	

Primitivos

abastract	**boolean**	break	**byte**	case
catch	**char**	class	const	continue
default	do	**double**	else	extends
final	finally	**float**	for	goto
if	implements	import	instanceof	**int**
interface	**long**	native	new	package
private	protected	public	return	**short**
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while	assert	

Sin Uso

abastract	boolean	break	byte	case
catch	char	class	**const**	continue
default	do	double	else	extends
final	finally	float	for	**goto**
if	implements	import	instanceof	int
interface	long	native	new	package
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while	assert	

Rangos

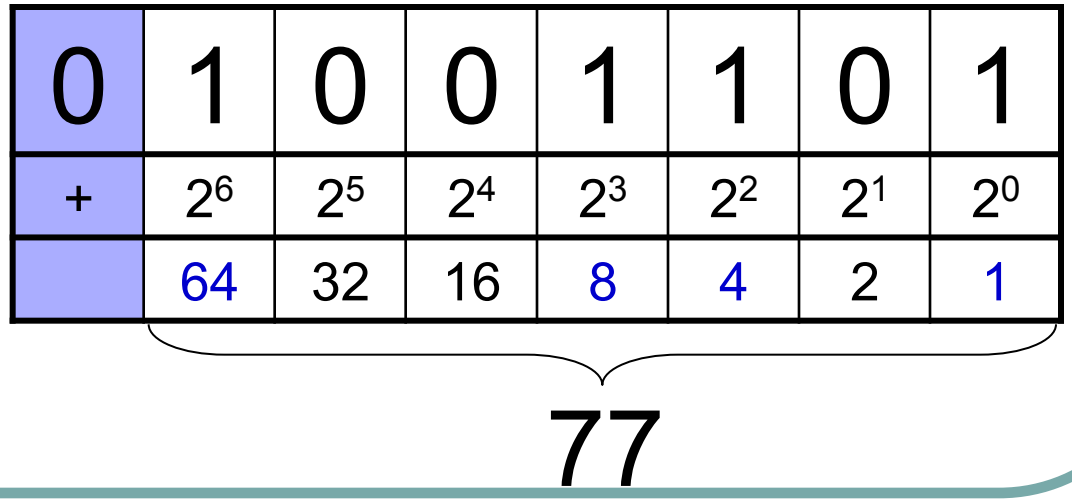
Tipo	Bits	Byte	Rango mínimo	Rango máximo
byte	8	1	-2^7	2^7-1
short	16	2	-2^{15}	$2^{15}-1$
int	32	4	-2^{31}	$2^{31}-1$
long	64	8	-2^{63}	$2^{63}-1$
float	32	4	---	---
double	64	8	---	---

Primitivos numéricos

- Todos tienen signo

Signo
0 : Positivo
1 : Negativo

byte x = 77



Otras formas de expresar números

- Octal (Base 8)
  - int seis = 06;
  - int siete = 00007;
  - int ocho = 010;
- Hexadecimal (Base 16)
  - int nueve = 0x9;
  - int diez = 0XA;
  - int once = 0X000b;

Ejemplo: Numeros.java

long, float, double

- long num = 130;
- long num = 150L;
- float num = 19;
- float num = 1.2f; // F
- double num = 30.7;
- double num = 12e4;
- double num = 12.34e56d;

Otros literales, boolean, char, String

- Boleanos
  - boolean flg = true; // o false
- Char
  - char a = ‘a’;
  - char nuevaLinea = ‘\n’;
  - char tablaUnicode = ‘\u0000’;
- String (el objeto especial)
  - String miLetra = “A”;

Ejemplo: Caracteres.java

Arreglos – Declaración

- Una dimension (primitivo)
  - int[] miArreglo; // preferido
  - int miArreglo[];
- Una dimension (objetos)
  - Thread[] misHebras; // preferido
  - Thread misHebras[];
- OJO: Error de compilación
  - Thread[5] misHebras;

Arreglos – Declaración

- Dos Dimenciones
  - int [] [] miArreglo;
  - int [] miArreglo [];
  - int miArreglo [] [];
- Multiples Dimenciones
  - String [] [] [] nombres; // tres dimenciones
  - String [] [] misHebras [] []; // cuatro dimen.

Arreglos – Inicialización

- int[] puntajes = new int[5];

0	0	0	0	0
0	1	2	3	4

- String nombres[] = new String[5];

null	null	null	null	null
0	1	2	3	4

Arreglos – Inicialización

- int[][] matris = new int[4][6];

matris[2][3] = 77;

idx	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	**77**	0	0
3	0	0	0	0	0	0

Ejemplo: Arreglos.java

Arreglos – Inicialización

- int[][] arregloDeArreglos = new int[2][];
- arregloDeArreglos[0] = new int [5];
- arregloDeArreglos[1] = new int [3];

arregloDeArreglos[1][3] = 77; //posición no existe

idx	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	x	x

Ejemplo: Arreglos.java

Arreglos – Inicialización en Línea

- int x = 8;
- int[] arr = {4, 6, x, 8}; //sólo en línea

- String[][] arr = { {“0 , 0”, “0 , 1”},
  {“1,0”, null, null, null}};

- Object[3] arr = {null, null, null}; //error

Ejemplo: Arreglos.java

Arreglos – Anónimos

- int [] arr;
- arr = new int[]{4, 6, 9, 8};
- arr = new int[]{10, 12, 14, 16};

- arr = new int[3]{0, 1, 2}; // error

- String a[];
- a = new String[]{null, new String("a")};

Ejemplo: Arreglos.java

Arreglos – Interfaz como tipo

```
interface Manejar{
    void conducir();
}
class Ferrari implements Manejar {
    void conducir(){ … }
}
class Honda implements Manejar {
    void conducir(){ … }
}
class Mercedes {
    void conducir(){ … }
}
```

- Manejar[] autos = new Manejar[3];
- autos[0] = new Ferrari();
- autos[1] = new Honda();
- autos[2] = new Mercedes(); // error, no implementa la interfaz

Variable e Inicialización por Omisión

```
class Variables{
 //Variables numericas inicializa en 0 / 0.0
  private int edad;
  // Variables booleanas en false
  private boolean esVip;
  //Variables char en '\u0000';
  private char sexo;
  // Variables de referencia (y arreglos) en null
  private String nombre;

  public static void main(String[] args){
     Variables var = new Variables();
     System.out.println("nombre:" + var.nombre.toUpperCase());
  }
}
```

Ejemplo: Variables.java

Variable e Inicialización por Omisión

```
class VariablesLocales{
  private String nombre;

  public String getNombre(){
    String nombreLocal;
    if (nombre == null){
      nombreLocal = “Sin Nombre”;
    }

    return (nombre == null) ? nombreLocal : nombre;
  }

  public static void main(String[] args){
      System.out.println(“nombre:” + (new VariablesLocales()).getNombre());
  }
}
```

Ejemplo: VariablesLocales.java

Argumentos al Método main(…)

```
public class TestMain{
    public static void main(String [] argumentos){

            System.out.println(“El argumento es : ” + argumentos[0]);

    }
}
```

Ejecucion:
]$ java TestMain Hola

Salida por consola
El argumento es : Hola

Argumentos al Método main(…)

```
public class TestMain{
    public static void main(String [] argumentos){

            System.out.println(“Nro Argumentos: ” + argumentos.length);

    }
}
```

Ejecución:
]$ java TestMain 1 2 3 4 5 6 7 8 9 0

Salida por consola
Nro Argumentos: 10