Asignaciones

Apilar y montar

Para la mayoría de las personas comprender los conceptos básicos de la pila y el montón lo hace mucho más fácil de entender temas como el paso de argumentos, el polimorfismo, los hilos, las excepciones y la recolección de basura. En esta sección, nos limitaremos a una visión general, pero ampliaremos estos temas varias veces más a lo largo del libro.

En su mayor parte, las diversas piezas (métodos, variables, y objetos) de los programas de Java viven en uno de los dos lógares en la memoria: la pila o el montón

Literales, Asignaciones y variables

Una literal primitiva es meramente una representación de fuente de código de los tipos de datos primitivos, en otras palabras, un entero, un punto flotante, booleano o caracter que escribes en el código. Lo siguiente son ejemplos de literales primitivas

```
'b' // char literal
42 // int literal
false // boolean literal
2 546789. 343 // double literal
```

Enteros literales

Hay tres formas de representar números enteros en el lenguaje Java: decimal (base 10), octal (base 8) y hexadecimal (base 16)

Decimales literales

Enteros decimales no necesita explicaciones; los has estado usado desde un grado uno o anterior.

Literales octales

Enteros octagonales solo usan los digitos del 0 al 7. En Java, puedes representar un entero en formal octal poniendo un Cero de lado izquierdo del número

Literales hexadecimales

Hexadecmal (hex para short) son construidos usando 16 distintos simbolos. Porque nunca inventó un sólo dígito simbólico para los números del 10 al 15, se usa caracteres alfabéticos para representar esos digitos.

Java acaptará letras mayúscolas o minúscolas para los dígitos adicionales Se le permite hasta 16 dígitos en un número hexadecimal, sin incluir el prefijo 0x o la extensión del sufijo opcional L, que se explicará más adelante

Para el caso de un long en hexadecimal, se pone una 'L' o 'l' dependiendo del caso, si no tiene el 0x, se pone la 'L' al final del número, del caso contrario la 'l'

Punto flotantes literales

Los números de punto flotante se definen como un número, Un símbolo decimal y más números que representan la fracción

Literales booleanas

Un valor booleano solo puede ser definido como verdadero o falso. Aunque en C (y algunos otros lenguajes) esto es común usar números para representar true or false, esto no funcionará en Java

Caracteres literales

Un caracter literal es representado por un solo caracter en comillas simples.

```
char a = 'd';
char b = '@';
```

Tambien puede escribir el valor Unicode del caracter, usando la notación Unicode de prefijar el valor \u de la siguiente manera:

```
char letterN = '\u004E'; // La letra 'N'
```

Recuerde, los caracteres son solo enteros sin signo de 16 bits debajo del capó

Valores literales para Strings

Un string es una representacion de una fuente de código de valores de objetos String

Operadores de asignación

Asignar un valor a una variable parece bastante sencillo, simplemente asigna las cosas en el lado derecho de = a la variable a la izquierda Asignar valor a una variable parece bastente sencillo, simplemente asigna las cosas en el lado derecho de = a la variable de izquierda

Asignamientos primitivos

El signo igual (=) se usa para asignar un valor a una variable y se denomina inteligentemente operador de asignación. En realidad, hay 12 operadores de asignación, pero solo cinco más comúnmente utilizados.

```
int x = 7;
int y = x+2;
int z = x*y;
```

Fundición primitiva

Los casts pueden ser implícitos o explícitos. Una conversión implícita significa que no tiene que escribir código para la conversión. La conversión de gran valor en contenedor pequeño se conoce como estrechamiento y requiere una conversión explícita, donde le dice al compilador que está consciente del peligro y acepta la responsabilidad total.

Conversión implícita

```
int a = 100;
long b = a; // conversión implícita
```

Conversión explícita

```
float a = 100.001f;
int b = (int) a; // Reparto explícito, el flotada podría
                 // perder información
```

Asignación de números punto-flotantes

Los números de punto flotante tienen un comportamiento de asignación ligeramente diferente al de los tipos enteros. Primero, debe saber que cada literal de punto flotante es implícitamente un doble (64 bits), no un flotante. Las siguientes asignaciones compilarían.

```
float f = (float) 32.3;
float g = 32.3f;
float h = 32.3f;
```

Asignar una literal que es demasiado grande para la variable

Si intentamos asignar un valor literal que es demasiada grande para la variable, tendríamos un error en el compilador

```
byte a = 128; // byte solo puede valores mínimos de 127
```

El presente código da un error algo como

```
TestBytes.java:5: posible loss of precisión
found   : int
required: byte
byte d = 128;
```

podemos componerlo con un casteo

```
byte a = (byte) 128;
```

Asignación Una variable primitiva a otra variable primitiva

La asignación de una variable hacia otra consiste en asignar el valor de una hacia la otra como una copia de esta misma

Asignación Variables de Referencia

Puede asignar un objeto recien creado a una variable de referencia de objeto de la siguiente manera:

```
Boton b = new Boton();
```

La línea anterior hace tres cosas clave:

- Hace una variable de referencia llamada b, de tipo botón.
- Crea un nuevo objeto botón en el montón
- Asigna el objeto botón recien creado a la variable de referencia b

Puede además asignar null para un objeto de variable de referencia, cual simplemente significa que la variable no esta refereciando a algún objeto:

```
Boton c = null;
```

Variable de alcanze

Son aquellas variables que se encuentra en una clase.

Usando una variable o Elemento Array que es inasignado y sin inicializar

Java nos da la opción de inicializar una variable declarada o dejarla sin inicializar. Cuando intentamos utilizar la variable sin inicializar, podemos obtener un comportamiento diferente según el tipo de variable o matriz con la que tratamos (primitivos u objetos). El comportamiento también depende del nivel (alcance) en el que estamos declarando nuestra variable. Una variable de instancia se declara dentro de la clase pero fuera de cualquier método o constructor, mientras que una variable local se declara dentro de un método (o en la lista de argumentos del método).

Variables de instancia primitivas y de tipo objeto

Las variables de instancia (también llamadas variables miembro) son variables definidas en el nivel de clase. Eso significa que la declaración de variable no se realiza dentro de un método, constructor o cualquier otro bloque de inicialización. Las variables de instancia se inicializan a un valor predeterminado cada vez que se crea una nueva instancia, aunque se les puede dar un valor explícito una vez que se hayan completado los superconstructores del objeto.

Tipo de variable	valor por defecto
referencia de objeto	null (no referencia a ningún objeto)
byte, short, int, long	0
float, double	0.0
boolean	false
char	'\u0000'

Variables primitivas de instancia

Son las variables definidas dentro de una clase como las características que posee el objeto y en donde se utilizan los tipos de datos primitivos.

Variables de instancia de referencia de objeto.

Son las variables definidas dentro de una clase como las características que posee el objeto y en donde se utilizan otros objetos

Variables de instancia de arreglos

Una matriz es un objeto; por lo tanto, una variable de instancia de matriz que se declara pero no se inicializa explícitamente tendrá un valor nulo, al igual que cualquier otra variable de instancia de referencia de objeto

Referencia de objetos locales

Las referencias de los objetos también se comportan de manera diferente cuando se declaran dentro de un método en lugar de como variables de instancia.

Con la referencia de objeto de variable de instancia puede salirte sin dejar una referencia de objeto sin inicializar.

Variable de referencia de instancia siempre dan un valor nulo, hasta que este se inicialize. Pero una referencia local no da un valor por defecto; en otras palabras no se inicializan en null.

Arreglos locales

Al igual que calquier otra referencia de objeto, a las referencias de matriz declaradas dentro de un método se les debe asignar un valor antes de usar. Eso sólo significa que debe declarar y construir la matriz. Sin embargo, no es necesario inicializar explícitamente los elementos de una matriz.

Asignar una variable de referencia a otra

Con las variables primitivas; una asignación de una variable a otra significa que los contenidos (patrón de bits) de una variable se copian de otra

Pasando variables a métodos

Pasando variables de objeto de referencia

Cuando paso una variable de objeto a un método, debe tener en cuenta que está pasando la referencia del objeto y no el objeto en sí.

Una copia de una variable significa que obtiene una copia de los bits en esa variable, por lo que cuando pasa una variable de referencia, está pasando una copia de los bits, que representan cómo llegar a un objeto específico

Pasando variables primitivas

En este caso de paso de variables primitivas si se realiza una copia de la variable de lo cual protege la integridad de esta misma sin que alterar el valor inicial de la que le fue asignado.