Type Casting, Array And Operators and Precedence

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

Type Casting?

- Assigning a value of one type to a variable of another type is known as **Type Casting**.

Categories of Type Casting

- In Java, type casting is classified into two types,
- Widening Casting(Implicit)

- Narrowing Casting(Explicitly done)

Widening or Automatic type conversion

Automatic Type casting take place when,

- the two types are compatible
- the target type is larger than the source type

Narrowing or Explicit type conversion

- When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.

```
Explicit casting
   int i = (int)3.0; (type narrowing)
   int i = (int)3.9; (Fraction part is
  truncated)
```

Example

```
public class Test {
 public static void main(String[] args) {
double d = 100.04;
 long l = (long)d; //explicit type casting required
int i = (int)l; //explicit type casting required
System.out.println("Double value "+d);
System.out.println("Long value "+l);
System.out.println("Int value "+i);
} }
```

O/P:

Double value 100.04

Long value 100

Int value 100

Example

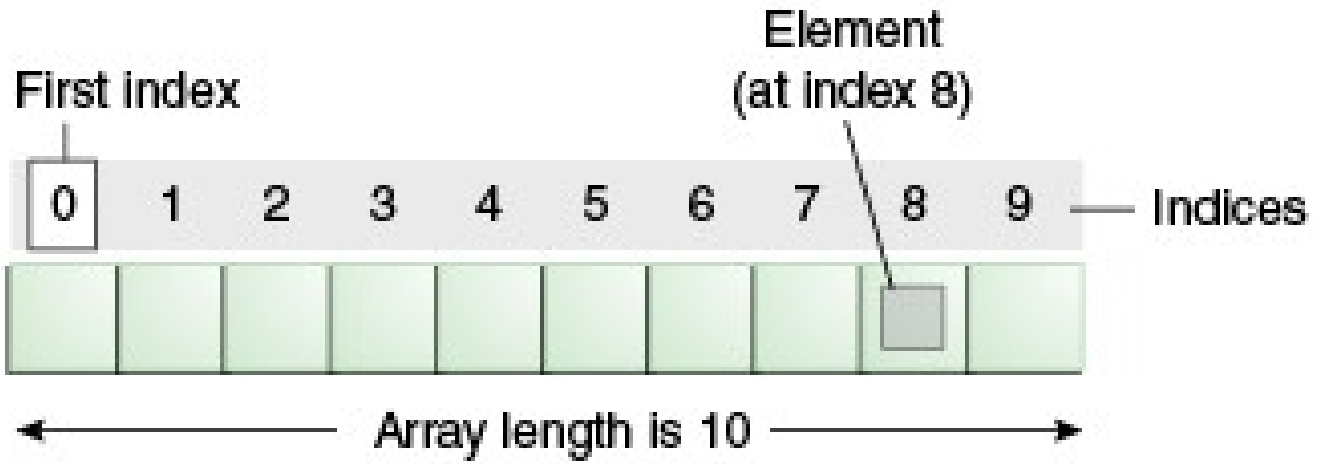
```
public class Test {
public static void main(String[] args) {
    int i = 100; long l = i; //no explicit type casting required
    float f = l; //no explicit type casting required
        System.out.println("Int value "+i);
        System.out.println("Long value "+l);
        System.out.println("Float value "+f);
}
}
```

o/p:

```
Int value 100
 Long value 100
Float value 100.0
```

Array?

- An *array* is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created. After creation, its length is fixed.



Array Declaration?

Syntax:

datatype[] identifier;

or

datatype identifier[];

- Array is a container object that hold values of homogenous type. It is also known as static data structure because size of an array must be specified at the time of its declaration.
- Index of array starts from zero to size-1.

Array Initialization ?

Syntax:

```
int[] arr = new int[10]; //10 is the size of array.
```

or

```
int[] arr = {10,20,30,40,50};
```

- new operator is used to initialize an array.

Accessing array element?

- As mention earlier array index starts from 0. To access nth element of an array.

Syntax

arrayname[n-1];

Example : To access 4th element of a given array

```
int[] arr={10,20,30,40};
System.out.println("Element at 4th place"+arr[3]);
```

foreach or enhanced for loop

- J2SE 5 introduces special type of for loop called foreach loop to access elements of array. Using foreach loop you can access complete array sequentially without using index of array. Let us see an exapmle of foreach loop.

Example of foreach

```
class Test {
 public static void main(String[] args) {
int[] arr={10,20,30,40};
 for(int x:arr) {
System.out.println(x);
 }
}
}
```

output: 10
20
30
40

Example of foreach

```
class Test {
 public static void main(String[] args) {
int[] arr={10,20,30,40};
 for(int x:arr) {
System.out.println(x);
 }
}
}
```

output: 10
20
30
40

JAVA OPERATORS

Operators

- Java provides a rich set of operators environment. Java operators can be divided into following categories
- Arithmetic operators
- Relation operators
- Logical operators
- Bitwise operators
- Assignment operators
- Conditional operators
- Misc operators

Arithmetic operators

- Arithmetic operators are used in mathematical expression in the same way that are used in algebra.

operator	description
+	adds two operands
-	subtract second operands from first
*	multiply two operand
/	divide numerator by denumerator
%	remainder of division
++	Increment operator increases integer value by one
--	Decrement operator decreases integer value by one

Relation operators

- The following table shows all relation operators supported by Java.

operator	description
==	Check if two operand are equal
!=	Check if two operand are not equal.
>	Check if operand on the left is greater than operand on the right
<	Check operand on the left is smaller than right operand
>=	check left operand is greater than or equal to right operand
<=	Check if operand on left is smaller than or equal to right operand

Logical operators

- Java supports following 3 logical operator. Suppose a=1 and b=0;

operator	description	example
&&	Logical AND	(a && b) is false
||	Logical OR	(a || b) is true
!	Logical NOT	(!a) is false

Bitwise operators

- Java defines several bitwise operators that can be applied to the integer types long, int, short, char and byte

operator	description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	left shift
>>	right shift

Assignment operators

- Assignment operator supported by Java are as follows

operator	description	example
=	assigns values from right side operands to left side operand	a=b
+=	adds right operand to the left operand and assign the result to left	a+=b is same as a=a+b
-=	subtracts right operand from the left operand and assign the result to left operand	a-=b is same as a=a-b
*=	mutiply left operand with the right operand and assign the result to left operand	a*=b is same as a=a*b
/=	divides left operand with the right operand and assign the result to left operand	a/=b is same as a=a/b
%=	calculate modulus using two operands and assign the result to left operand	a%=b is same as a=a%b

Conditional operator

- It is also known as ternary operator and used to evaluate Boolean expression

Syntax:

epr1 ? expr2 : expr3

- If **epr1** Condition is true? Then value **expr2** : Otherwise value **expr3**

Example

```
class ComparisonDemo {
public static void main(String[] args){
int value1 = 1;
 int value2 = 2;
if(value1 == value2)
 System.out.println("value1 == value2");
if(value1 != value2)
System.out.println("value1 != value2");
if(value1 > value2)
 System.out.println("value1 > value2");
 if(value1 < value2)
System.out.println("value1 < value2");
if(value1 <= value2)
System.out.println("value1 <= value2");
 }
}
```

Output:

```
value1 != value2
value1 < value2
value1 <= value2
```

Example

```
class ConditionalDemo1 {
public static void main(String[] args){
 int value1 = 1;
int value2 = 2;
if((value1 == 1) && (value2 == 2))
System.out.println("value1 is 1 AND value2 is 2");
if((value1 == 1) || (value2 == 1))
System.out.println("value1 is 1 OR value2 is 1");
 }
 }
```

Example

```
class ConditionalDemo2 {
public static void main(String[] args){
int value1 = 1;
int value2 = 2;
 int result;
 boolean someCondition = true;
result = someCondition ? value1 : value2;
System.out.println(result);
 }
}
```

Thank you