Java Programming AP Edition
U1C3 Basic Java Application Programming Interface

BOOLEAN DATA TYPE

ERIC Y. CHOU, PH.D.   IEEE SENIOR MEMBER

Java boolean values boolean constants

Only two boolean value: **true**, **false**.

boolean constants:

Very useful for use in program conditional code or compilation configurations.

- private static final boolean YES = **true**;
- private static final boolean NO = **false**;
- private static final boolean DEBUG = **true**;

boolean function (methods)

```java
public class Test {

  public static void main(String args[]) {
    System.out.println(Character.isLetter('c'));
    System.out.println(Character.isLetter('5'));
  }
}
```

Java boolean Expressions

A basic Boolean expression has this form:

expression relational-operator expression

Java evaluates a Boolean expression by first evaluating the expression on the left, then evaluating the expression on the right, and finally applying the relational operator to determine whether the entire expression evaluates to true or false.

The `boolean` Type and Operators

Often in a program you need to compare two values, such as whether i is greater than j. Java provides six comparison operators (also known as relational operators) that can be used to compare two values. The result of the comparison is a Boolean value: true or false.

```
boolean b = (1 > 2);
```

Boolean Data Type

The Boolean data type declares a variable with the value either true or false.

Relational Operators				
Java Operator	Math Symbol	Name	Example	Result
<	$<$	Less than	radius < 0	false
<=	$\leq$	Less than or Equal to	radius <= 0	false
>	$>$	Greater than	radius > 0	true
>=	$\geq$	Greater than or equal to	radius >= 0	true
==	$=$	Equal to	radius == 0	false
!=	$\neq$	Not Equal to	radius != 0	true

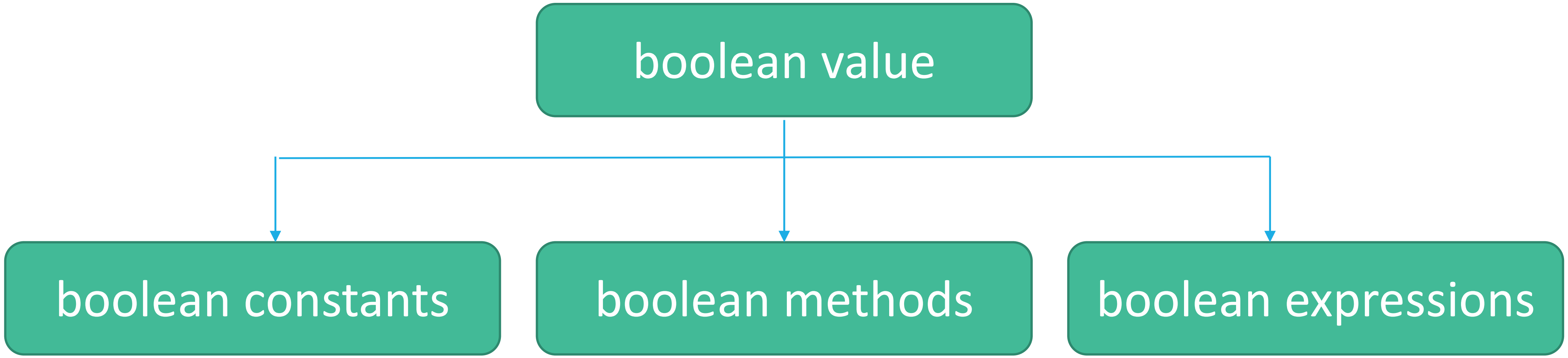
Boolean literals: **true** and **false.** These are the only values that will be returned by the Boolean expressions.

Java boolean Expressions

For example, suppose you have declared two variables: **int i = 5;** **int j = 10;**

Expression	Value	Explanation
`i == 5`	`true`	The value of `i` is `5`.
`i == 10`	`false`	The value of `i` is not `10`.
`i == j`	`false`	`i` is `5`, and `j` is `10`, so they are not equal.
`i == j - 5`	`true`	`i` is `5`, and `j - 5` is `5`.
`i > 1`	`true`	`i` is `5`, which is greater than `1`.
`j == i * 2`	`true`	`j` is `10`, and `i` is `5`, so `i * 2` is also `10`.

boolean values



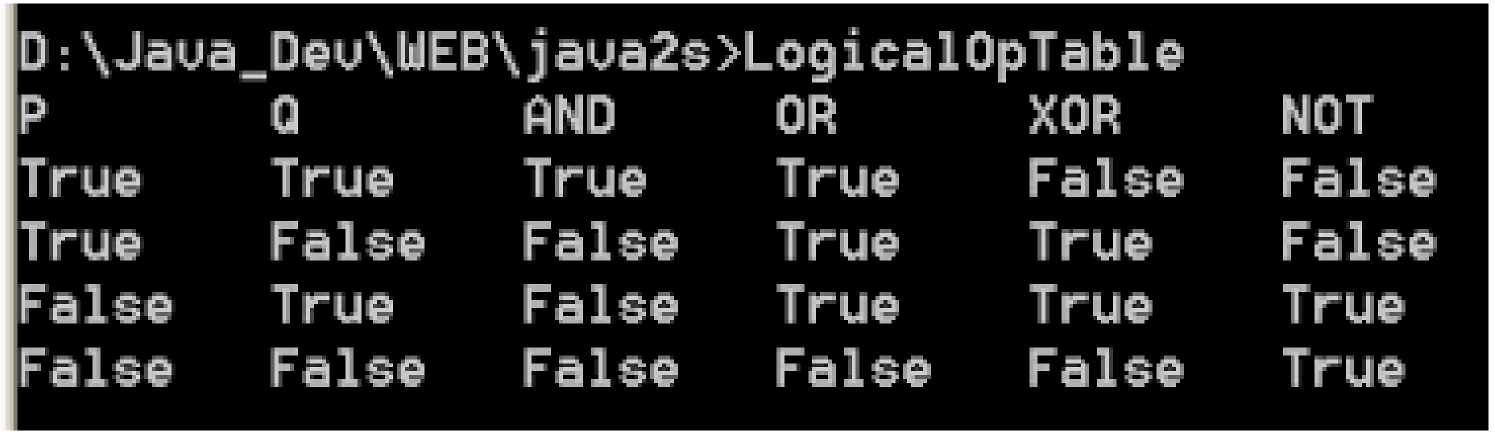
true
final boolean YES = **true**;

isLetter()
isLowerCase()

a >= 3

Logical Operators for Implementation of Boolean Logic

Boolean Operators		
Operator	Name	Description
!	not	Logical negation
&&	and	Logical conjunction
||	or	Logical disjunction
^	exclusive or	Logical exclusion (non-AP)



```
D:\Java_Dev\WEB\java2s>LogicalOpTable
P       Q       AND     OR      XOR     NOT
True    True    True    True    False   False
True    False   False   True    True    False
False   True    False   True    True    True
False   False   False   False   False   True
```

Truth Table for Operator !

p	!p	Example (assume age = 24, gender = 'M')
true	false	!(age > 18) is false, because (age > 18) is true.
false	true	!(gender != 'M') is true, because (grade != 'M') is false.

Truth Table for Operator &&

p1	p2	p1 && p2	Example (assume age = 24, gender = 'F')
false	false	false	(age > 18) && (gender == 'F') is true, because (age > 18) and (gender == 'F') are both true.
false	true	false	
true	false	false	(age > 18) && (gender != 'F') is false, because (gender != 'F') is false.
true	true	true	

Truth Table for Operator ||

p1	p2	p1 || p2	Example (assume age = 24, gender = 'F')
false	false	false	(age > 34) || (gender == 'F') is true, because (gender == 'F') is true.
false	true	true	
true	false	true	(age > 34) || (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false.
true	true	true	

Truth Table for Operator ^

p1	p2	p1 ^ p2	Example (assume age = 24, gender = 'F')
false	false	false	(age > 34) ^ (gender == 'F') is true, because (age > 34) is false but (gender == 'F') is true.
false	true	true	
true	false	true	(age > 34) || (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false.
true	true	false	

boolean data application

<condition> if a decision box

```
boolean wartime = true;
if (a.gender.isMale() && (a.age <=25 && a.age >= 18) &&
wartime){
        armyDraft(a);
   }
```

Look at part3: AP Exam Taking Skills (Boolean Logic)

For extra info: Boolean Wrapper Class.

Boolean class (non-AP)

(Wrapper Class for boolean)

The **Boolean** class wraps a value of the primitive type boolean in an object. An object of type Boolean contains a single field whose type is **boolean**.

In addition, this class provides many methods for converting a boolean to a String and a String to a boolean, as well as other constants and methods useful when dealing with a boolean.

Attributes and Methods in Boolean

Constants:

Boolean.TRUE

Boolean.FALSE

Modifier and Type	Method and Description
boolean	booleanValue()Returns the value of this Boolean object as a boolean primitive.
static int	compare(boolean x, boolean y)Compares two boolean values.
int	compareTo(Boolean b)Compares this Boolean instance with another.
boolean	equals(Object obj)Returns true if and only if the argument is not null and is a Boolean object that represents the same boolean value as this object.
static boolean	getBoolean(String name)Returns true if and only if the system property named by the argument exists and is equal to the string "true".
int	hashCode()Returns a hash code for this Boolean object.
static int	hashCode(boolean value)Returns a hash code for a boolean value; compatible with Boolean.hashCode().
static boolean	logicalAnd(boolean a, boolean b)Returns the result of applying the logical AND operator to the specified boolean operands.
static boolean	logicalOr(boolean a, boolean b)Returns the result of applying the logical OR operator to the specified boolean operands.
static boolean	logicalXor(boolean a, boolean b)Returns the result of applying the logical XOR operator to the specified boolean operands.
static boolean	parseBoolean(String s)Parses the string argument as a boolean.
String	toString()Returns a String object representing this Boolean's value.
static String	toString(boolean b)Returns a String object representing the specified boolean.
static Boolean	valueOf(boolean b)Returns a Boolean instance representing the specified boolean value.
static Boolean	valueOf(String s)Returns a Boolean with a value represented by the specified string.