AP Computer Science A

Lecture note 02

Student: _____ Instructor: 杜博识 Dubos Date: _____

Relational operators

Relational operators 关系算子	• Relational operators are used in **boolean expressions** that evaluate to `true` or `false` • Be careful with comparing double variables since there might be round-off error • Relational operators should generally be used only in the comparison of int and boolean. Objects are compared using the `equal` and `compareTo` methods.	
`>`	Greater than • `5 > 0` is boolean with value of `true`	
`<`	Less than • `0 < 5` is boolean with value of `true`	
`>=`	Greater than or equal to • `0 >= 0` is boolean with value of `true`	
`<=`	Less than or equal to • `0 <= 0` is boolean with value of `true`	
`==`	Equal to • `5 == 10 / 2` is boolean with value of `false` •	
`!=`	Not Equal to • `5 != 0` is boolean with value of `true`	

Logical operators

Logical operators 逻辑算子	• Logical operators are applied to **boolean expressions** that evaluate to `true` or `false`	
`!`	• Not • `! true` is `false` • `! false` is `true`	
`&&`	• And • `true && true` is `true` • `true && false` is `false` • `false && true` is `false` • `false && false` is `false` • **Short-circuit evaluation**: the code after `false &&` will	

	not be executed since the result of the compound Boolean expression will always be `false`	
`||`	• Or • `true || true` is `true` • `true || false` is `true` • `false || true` is `true` • `false || false` is `false` • **Short-circuit evaluation**: the code after `true ||` will not be executed since the result of the compound Boolean expression will always be `true`	

Operator precedence

Operator precedence 算符优先级	From higher to lower: • Logic `!` • Arithmetic `* / %` • Arithmetic `+ -` • Relational `> < >= <=` • Relational `== !=` • Logic `&& ||` • Assignment `= += -+ *= /= %=`	

De Morgan’ s Law

De Morgan’ s Law
德摩根律
	• Both `a` and `b` are Boolean
	`!(a && b)` is the same as `!a || !b`
	`!(a || b)` is the same as `!a && !b`	

if-else Statement

`if`	`if (/*condition in Boolean expression*/) {`
	`  //statements;`
	`}`
	• The statements will be executed only if the condition is `true`	

`if-else`	`if (/*condition1 in Boolean expression*/) {`
	`  //statements 1;`
	`} else {`
	`  //statements 2;`
	`}`
	• The statements 1 will be executed if condition1 is `true`
	• The statements2 will be executed if condition1 is `false`	

	• In nested or extended cases, note that `else` always gets matched with the nearest unpaired `if`	

Source code of LN02: https://github.com/Duboshi/Lecture/blob/main/src/LN02.java

```java
/*
 * Lecture note 02
 * Boolean, Logic, If-else statements
 */

public class LN02 {

    public static void main(String[] args) {

        //Relational operators
        boolean b = 5>0;
        p(b);
        p(0 < 5);
        p(0 >= 0);
        p(0 <= 0);
        p(5 == 10 / 2);            // arithmetic > relational
        p(5 != 0);
        p(0.3 == 0.1 + 0.1 + 0.1); //false, round-off error

        p("Ai".equals("Zeng"));    //false
        p("Ai".compareTo("Zeng")); //-25

        //Logical operators
        boolean t = true;
        boolean f = false;
        p(t || 5/0==2); //t or 5<0, after "true or" are dead code
        p(f && 5/0==2); //f and 5>0, after "false and" are dead code
        p(!t);  //not true

        p(!(t && f));   //true,
        p(!t && f);     //false, indicating: Logic! > Relational &&, ||

        //De Morgan's Law
        //1. !(a&&b) == !a || !b
        //2. !(a||b) == !a && !b

        //Operator precedence
        //Logic! > Arithmetic*/% > Arithmetic+- > Relational > Logic && || > Assignment
        int n = 6;
        p(!t || 3*(n/=2)==9);  //(not true) or (true), the result is true

        //if-else
        //Final Exam: A90, B80, C70, D60
        grade1(93); //A
        grade2(93); //A
        grade3(93); //A
        grade4(93); //A B C D
    }

    public static void p(Boolean b) {
        System.out.println(b);
    }

    public static void p(int n) {
        System.out.println(n);
    }

    public static void p(double x) {
        System.out.println(x);
    }

    public static void p(String s) {
        System.out.println(s);
    }

    public static void grade1 (int n) {
```

```
        //This is the standard solution
        if (n>=90) {
            System.out.println("A");
        } else {
            if (n>=80) {
                System.out.println("B");
            } else {
                if (n>=70) {
                    System.out.println("C");
                } else {
                    System.out.println("D");
                }
            }
        }
    }

    public static void grade2 (int n) {
        //The braces {} are optional for branches that have only one statement.
        //grade2 is exactly the same as grade1, with braces {} omitted,
        //since every branch has only one statement.
        if (n>=90)
            System.out.println("A");
        else
            if (n>=80)
                System.out.println("B");
            else
                if (n>=70)
                    System.out.println("C");
                else
                    System.out.println("D");
    }

    public static void grade3 (int n) {
        //grade3 is also the same as grade1, but with poor coding style.
        //grade3 may be used in the AP Exam as a challenge to the students.
        //In cases like this, every "else" corresponds to the "if" that is closest to it.
        if (n>=90) {
            System.out.println("A");
        } else if (n>=80) {
            System.out.println("B");
        } else if (n>=70) {
            System.out.println("C");
        } else {
            System.out.println("D");
        }
    }

    public static void grade4 (int n) {
        //grade4 is not the intended solution for n
        if (n>=90)
            System.out.println("A");
        if (n>=80)
            System.out.println("B");
        if (n>=70)
            System.out.println("C");
        if (n>=60)
            System.out.println("D");
    }

}
```