AP Computer Science A

Java Programming Essentials [Ver.4.0]

Unit 2: Selection and Iterations

CHAPTER 7: BOOLEAN ALGEBRA

DR. ERIC CHOU
IEEE SENIOR MEMBER

AP Computer Science Curriculum

- Comparing Boolean Expressions (T 2.6)
- Compound Boolean Expressions (T 2.5)

Objectives:

- Boolean Algebra
- Proof and Boolean Theorems
- Logic Design
- Boolean conditions, Compound Conditions: isEven, allEven, hasEven
- Calendar Year Project

Basic Boolean Algebra

Lecture 1

Introduction

- The most **obvious** way to **simplify** Boolean expressions is to manipulate them in the same way as normal **algebraic expressions** are manipulated.
- With regards to logic relations in digital forms, a set of rules for symbolic manipulation is needed in order to solve for the unknowns.

Introduction

- A set of rules formulated by the English mathematician George Boole describe certain propositions whose outcome would be either true or false.
- With regard to digital logic, these rules are used to describe circuits whose state can be either, 1 (**true**) or 0 (**false**). In order to fully understand this, the relation between the AND gate, OR gate and NOT gate operations should be appreciated. A number of rules can be derived from these relations as Table 1 demonstrates.

Introduction

- A number of rules can be derived from these relations as Table 1 demonstrates.

  - **Rule 1: X = 0 or X = 1**
  - **Rule 2: 0 * 0 = 0**
  - **Rule 3: 1 + 1 = 1**
  - **Rule 4: 0 + 0 = 0**
  - **Rule 5: 1 * 1 = 1**
  - **Rule 6: 1 * 0 = 0 * 1 = 0**
  - **Rule 7: 1 + 0 = 0 + 1 = 1**

Table 1: Boolean Postulates

Laws of Boolean Algebra

- Table 2 shows the basic Boolean laws.
- Note that every law has two expressions, (a) and (b). This is known as *duality*.
- These are obtained by changing every AND(*) to OR(+), every OR(+) to AND(*) and all 1's to 0's and vice-versa.
- It has become conventional to drop the * (AND symbol) i.e. A.B is written as AB.

Boolean Algebra Laws

T1 : Commutative Law

(a) $A + B = B + A$

(b) $A\,B = B\,A$

T2 : Associate Law

(a) $(A + B) + C = A + (B + C)$

(b) $(A\,B)\,C = A\,(B\,C)$

T3 : Distributive Law

(a) $A\,(B + C) = A\,B + A\,C$

(b) $A + (B\,C) = (A + B)\,(A + C)$

T4 : Identity Law

(a) $A + A = A$

(b) $A\,A = A$

T5 :

(a) $AB + A\overline{B} = A$

(b) $(A+B)(A+\overline{B}) = A$

T6 : Redundance Law

(a) $A + A\,B = A$

(b) $A\,(A + B) = A$

T7 :

(a) $0 + A = A$

(b) $0\,A = 0$

T8 :

(a) $1 + A = 1$

(b) $1\,A = A$

T9 :

(a) $\overline{A} + A = 1$

(b) $\overline{A}\,A = 0$

T10 :

(a) $A + \overline{A}\,B = A + B$

(b) $A\,(\overline{A} + B) = A\,B$

T11 : De Morgan's Theorem

(a) $(\overline{A + B}) = \overline{A}\;\overline{B}$

(b) $(\overline{A\,B}) = \overline{A} + \overline{B}$

True Table for Logic Functions

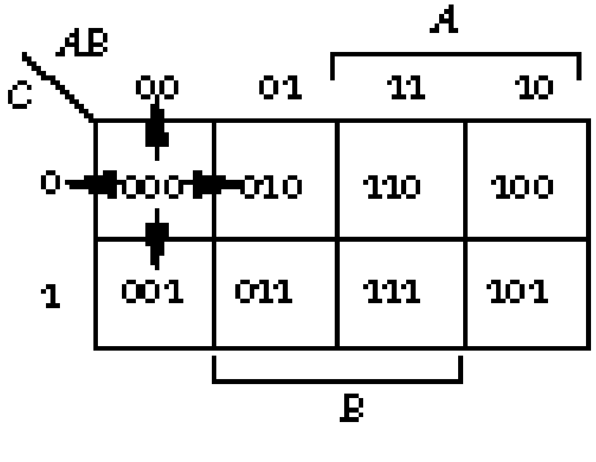
<expr A operator B> can have 16 possible outcomes which equals 2^4 (4 A/B combinations each has 2 possible outcome)

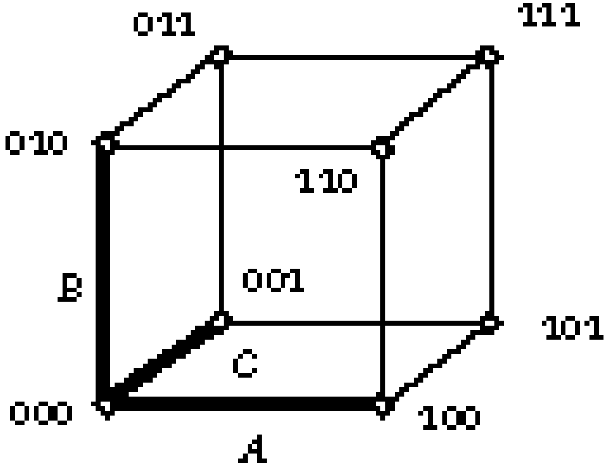
(Operators)

INPUT AB	ZERO	AND	A > B	A	B > A	B	XOR	OR	NOR	XNOR	NOT 'B'	A ≤ B	NOT 'A'	B ≤ A	NAND	ONE
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Cube and Theorem of Symmetry

The dual of any Boolean property





If Boolean Expression

T(A, B, +, *, =, 1, 0) is valid.

T(A, B , *, +, = 0, 1) is also valid.

A B + A = A

(A + B) * A = A

A + (-A) = 1

A * (-A) = 0

Example of De Morgan’s Theorem

- if ((x % 5 != 0) && (x % 2 != 0))   // x is not multiple of 5 and x is not multiple of 2.
- equivalent to
- if (!((x % 5 == 0) || (x % 2 == 0)))   // x is not (multiple of 5 or 2)

- Download and work on BooleanQuiz.pdf

Logic Design

Lecture 2

Student GPA Issues 1:

(smaller set should go first)

Smaller set go first:

```
if (math>=90)           {mathGrade = 'A';}
else if (math>=80)  {mathGrade = 'B';}
else if (math>=70)  {mathGrade = 'C';}
else if (math>=60)  {mathGrade = 'D';}
else                        {mathGrade = 'F';}
```

Smaller set go first:

```
if (math<60)            {mathGrade = ‘F';}
else if (math>=60){mathGrade = ‘D';}
else if (math>=70){mathGrade = 'C';}
else if (math>=80){mathGrade = ‘B';}
else if (math>=90){mathGrade = ‘A';}
```

Student GPA Issue 2:

(Corner Cases)

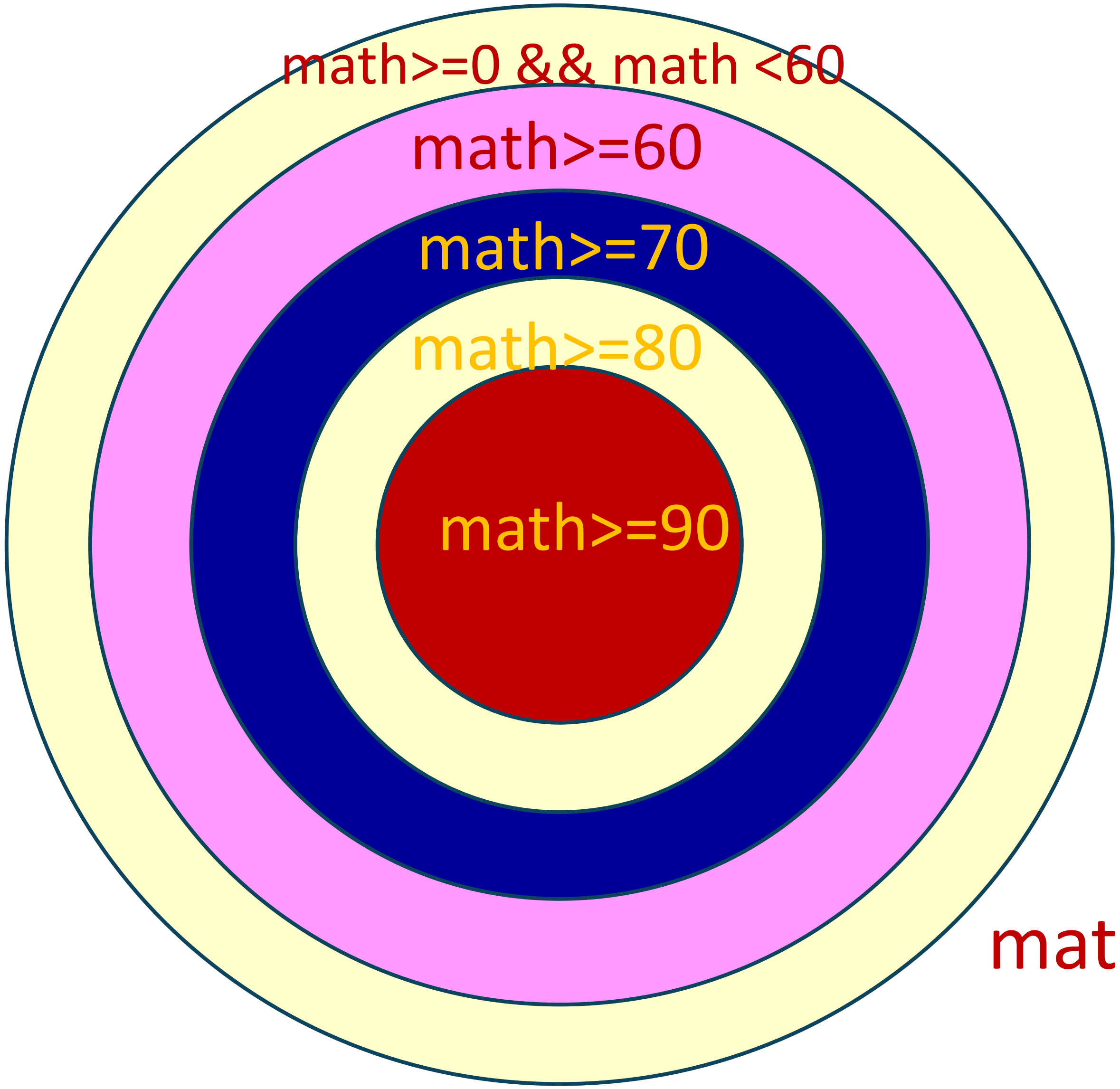
Smaller set go first:

```
if (math>=90)          {mathGrade = 'A';}
else if (math>=80)  {mathGrade = 'B';}
else if (math>=70)  {mathGrade = 'C';}
else if (math>=60)  {mathGrade = 'D';}
else                         {mathGrade = 'F';}
```

Corner Cases Handled:

```
 if (math>=90 && math<=100)
{mathGrade = 'A';}
 else if (math>=80 && math<=100)
{mathGrade = 'B';}
 else if (math>=70 && math<=100)
{mathGrade = 'C';}
 else if (math>=60 && math<=100)
{mathGrade = 'D';}
 else if (math>= 0 && math<=100)
{mathGrade = 'F';}
 else {mathGrade = 'N'; }  // not graded yet
```

Venn Diagram for StudentGPA series programs



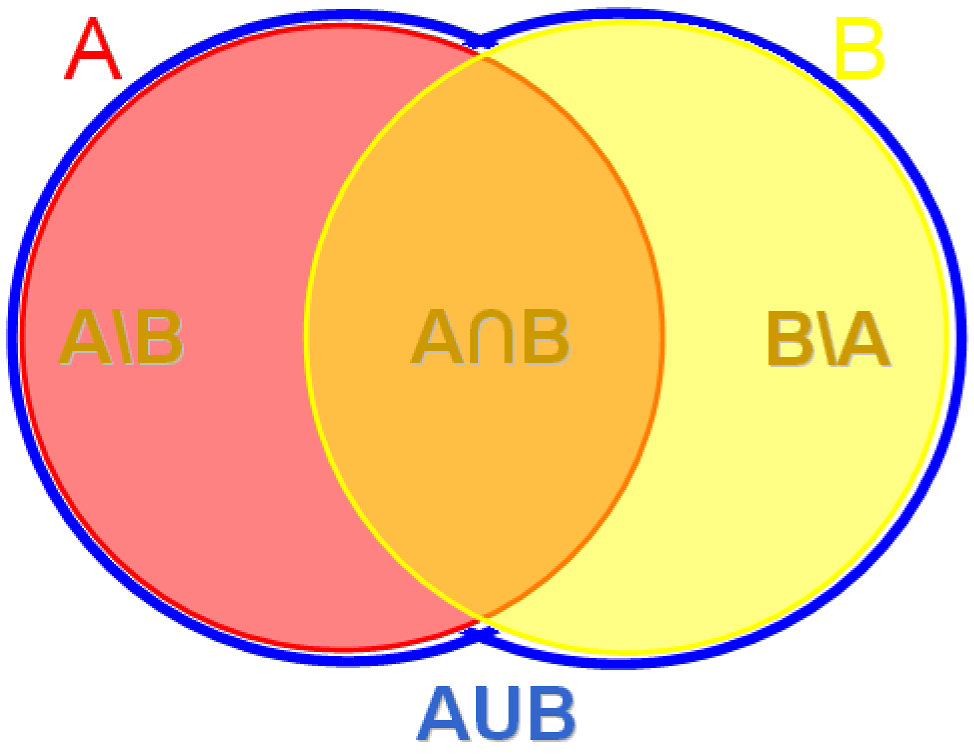
math>100 || math <0

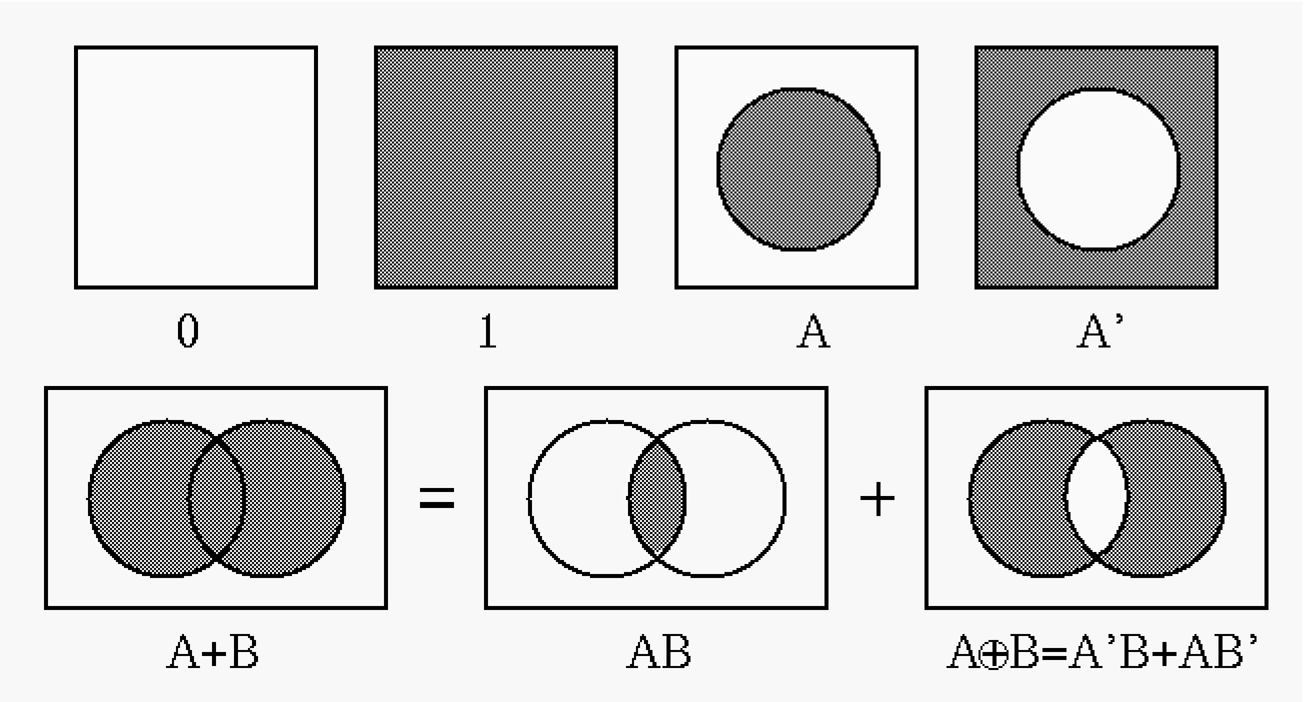
Set Theory and Logic Design (only positive numbers are shown)

- boolean m5 = (x % 5 == 0); // Set S5 = [0, 5, 10, 15, 20, ...]
- boolean m2 = (x % 2 == 0); // Set S2 = [0, 2, 4, 6, 8, 10, ...]

Boolean Expression	Set	Set Components
m5 && m2	$S5 \cap S2$	[0, 10, 20, 30, 40, ...]
m5 || m2	$S5 \cup S2$	[0, 2, 4, 5, 6, 8, 10, 12, 14, 15, ...]
m2 && !m5	$S2 - S5$	[2, 4, 6, 8, 12, 14, 16, 18, ...]
m5 && !m2	$S5 - S2$	[5, 15, 25, 35, 45, ...]
m2 ^ m5	$S5 \oplus S2$	[2,4, 5, 6, 8, 12, 14, 15, 16, ...]

Venn Diagram Analysis

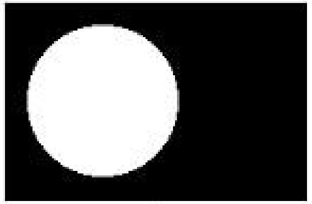


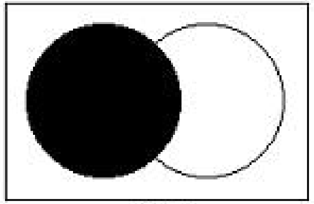


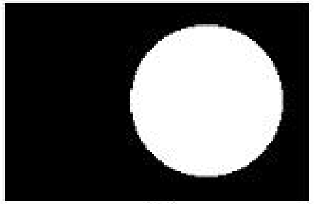
x	0	0	1	1
y	0	1	0	1
0	0	0	0	0
$x \cdot y$	0	0	0	1
$x \cdot y'$	0	0	1	0
x	0	0	1	1
$x' \cdot y$	0	1	0	0
y	0	1	0	1
$x \cdot y' + x' \cdot y$	0	1	1	0
$x+y$	0	1	1	1
$(x+y)'$	1	0	0	0
$x \cdot y + x' \cdot y'$	1	0	0	1
y'	1	0	1	0
$x+y'$	1	0	1	1
$x' \cdot y'$	1	1	0	0
$x'+y$	1	1	0	1
$(x \cdot y)'$	1	1	1	0
1	1	1	1	1

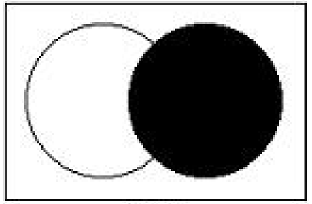
All 16 Boolean Functions can be expressed by (x, y, !, &&, ||)

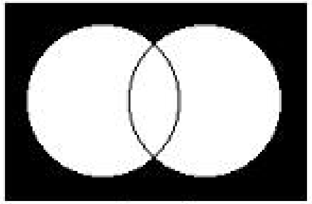
Boolean Functions in Java	
boolean f0 = false;	boolean f8 = !(x+y);
boolean f1 = x && y;	boolean f9 = x && y + !x && !y;
boolean f2 = x && !y;	boolean f10 = !y;
boolean f3 = x;	boolean f11 = x + !y;
boolean f4 = !x && y;	boolean f12 = !x && y;
boolean f5 = y;	boolean f13 = !x + y;
boolean f6 = x && !y + !x && y;	boolean f14 = !(x && y)
boolean f7 = x + y;	boolean f15 = true;

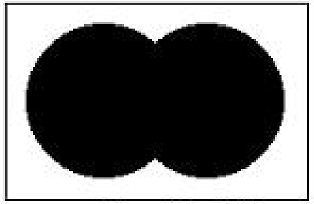

A

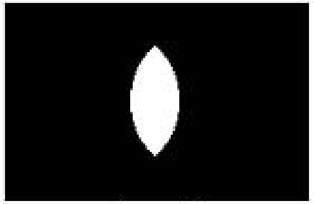

not A

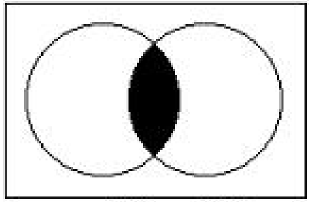

B

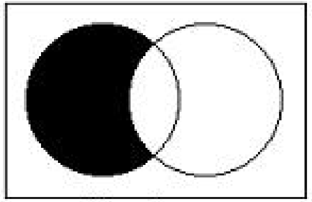

not B

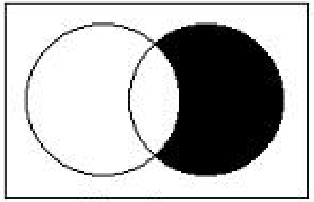

A or B

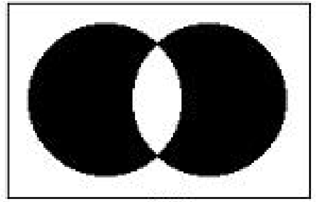

not A and not B

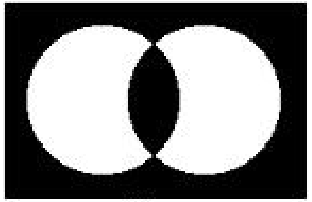

A and B

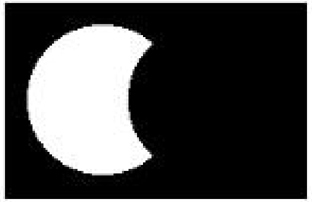

not (A and B)

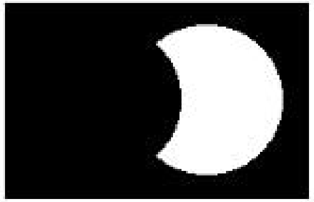

if A then B

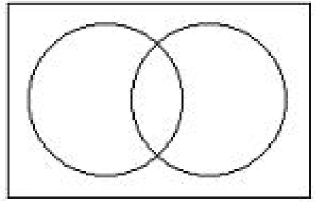

if B then A


A iff B


A iff not B


A and not B

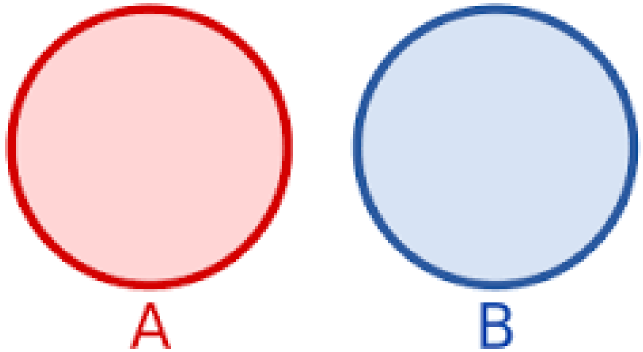

B and not A


Tautology


Contradiction

Mutual Exclusive Sets

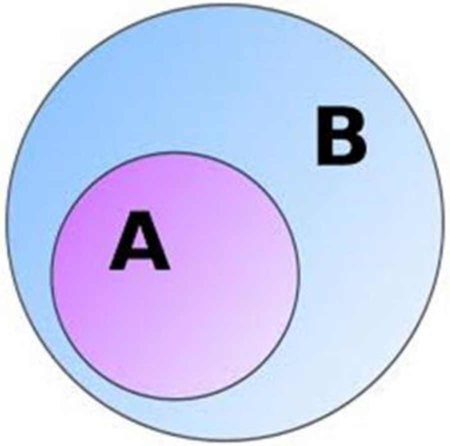
- boolean male   = (gender == ‘M’);
- boolean female = (gender == ‘F’);



- if (male) { // all male get here , and all female will not get here}
- if (female) { // all female get here, and all male will not get here}

Set Contained by Another Set

- Contained by (A == (x>2), B == (x>1))
- if a is true then b must be true   == !a || b
- If x > 2 then x > 1 == !(x > 2) || (x>1)   (x>2) is containec

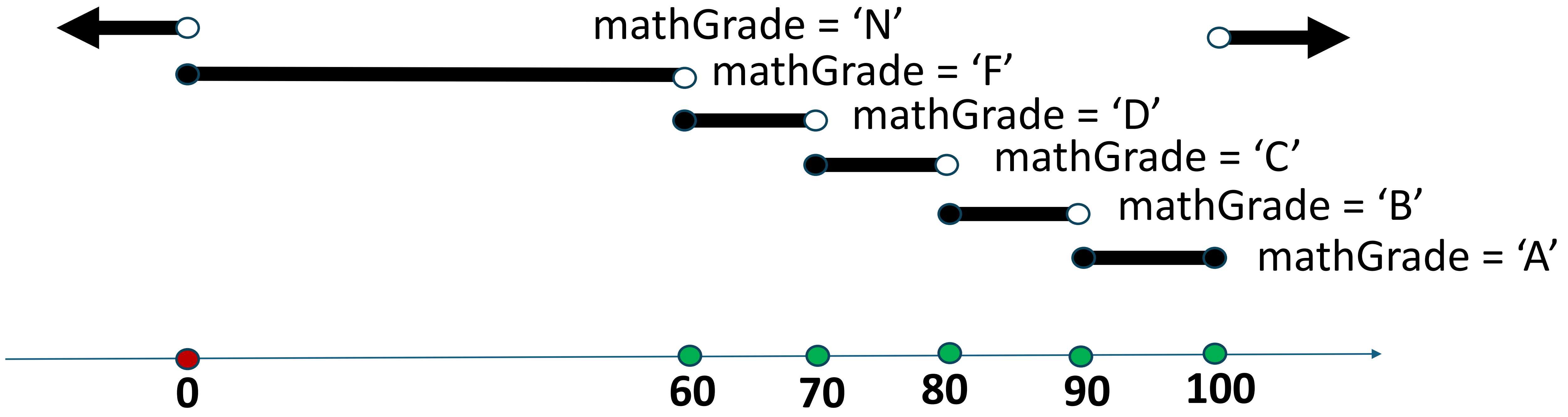


If you want to have A test first and B-A test:

```
if (x > 2) { // all (x>2) get here }
else if (x > 1) { // only x>1 && !(x>2) get here }
```

- If you want to have A test and then B test:
- **if (x>2) { // all x > 2 get here }**
- **if (x>1) { // all x > 1 get here }**
- Don’t try this: if (x>1) { // all x>1 get here } else if (x>2) { // no x can get here}

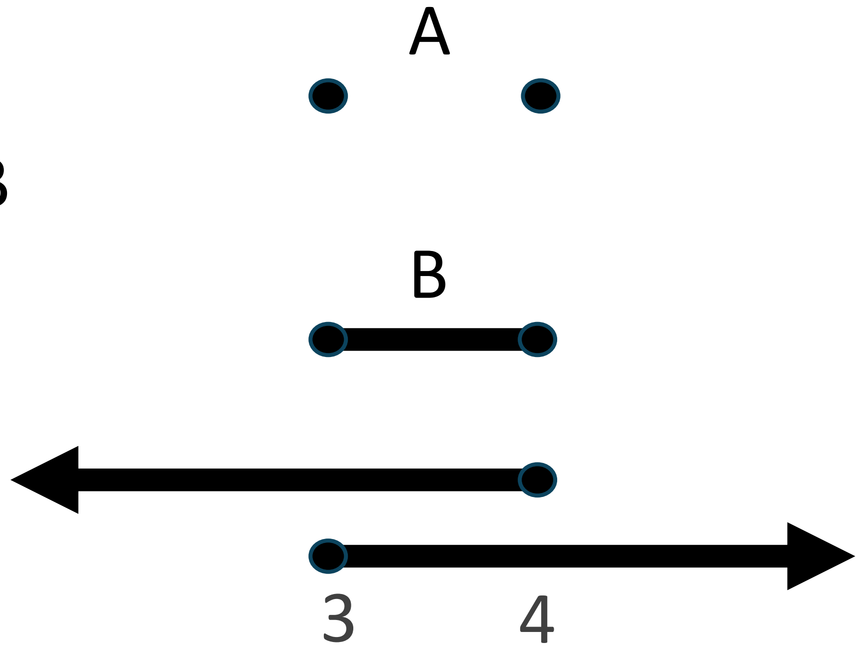
Number Line Analysis (Letter Grade)



Sometimes the Logic Design Result Also Depends on the Data Type as Well

- if (x == 3 || x == 4) { // set A }
- if (x >= 3 && x <= 4) { // set B }

- For int data type, A is equivalent to B
- For double data type, A is not equivalent to B
- for example, x = 3.5 is in B but not in A



Compound Logic

Lecture 3

Has Even

- Check if an array of integers has any even number.
- As long as there is an even number then reporting true.
- Otherwise, reporting false.

All Even

- Check if an array of integers are all even numbers.
- As long as there is an odd number then reporting false.
- Otherwise, reporting true.

Has Odd

- Check if an array of integers has any odd number.
- As long as there is an odd number then reporting true.
- Otherwise, reporting false.
- Has Odd is also equivalent to Not All Even.

All Odd

- Check if an array of integers are all odd numbers.
- As long as there is an even number then reporting false.
- Otherwise, reporting true.
- All Odd is also equivalent to Not Have Even.

Lab

Calendar Year

LeapYear.java

Background Information: (Leap Year)

- Which Years are Leap Years?
- In the Gregorian calendar 3 criteria must be take into account to identify leap years:

- The year is evenly divisible by 4;
- If the year can be evenly divided by 100, it is NOT a leap year, unless;
- The year is also evenly divisible by 400. Then it is a leap year.

- The year 2000 was somewhat special as it was the first instance when the third criterion was used in most parts of the world since the transition from the Julian to the Gregorian Calendar.

Year of the Rat

Year of the Ox

Year of the Tiger

Year of the Rabbit

Year of the Dragon

Year of the Snake

Year of the Horse

Year of the Sheep

Year of the Monkey

Year of the Rooster

Year of the Dog

Year of the Pig

Background Information:
(Chinese Zodiac)

- Jupiter goes around the sun every 11.87 years. (approximately 12 years.) Chinese call it ***Planet of Years***. They use it to calculate for the Zodiac. Every 12 years is one rotation.
- The year 1948 is year of Rat. The year 1950 is year of Tiger. The year 2015 is year of Sheep (sometimes also called goat/ram. Chinese believe they are in one category.)
- If the year 2015 is 67 (2015-1948) years away from 1948 and 67 % 12 is 7, 7 years away from rat year is year of Sheep.
- So, you may use y = (x-1948) % 12 to know that Year x is y years away from Rat year. Then, look up from the table.

Lab: Calendar Year (LeapYear.java)

- Write a program to ask the calendar year between 1948 and now, to determine 2 things:
  (1) Is it a leap year?
  (2) What Chinese Zodiac Year it is?
- Then, Print out a Calendar Year Report with these information.

Expected Result: