AP Computer Science A

Java Programming Essentials [Ver. 2.0]

Unit 1: Elementary Programming

WEEK 4: CHAPTER 3- BASIC JAVA API

DR. ERIC CHOU IEEE SENIOR MEMBER

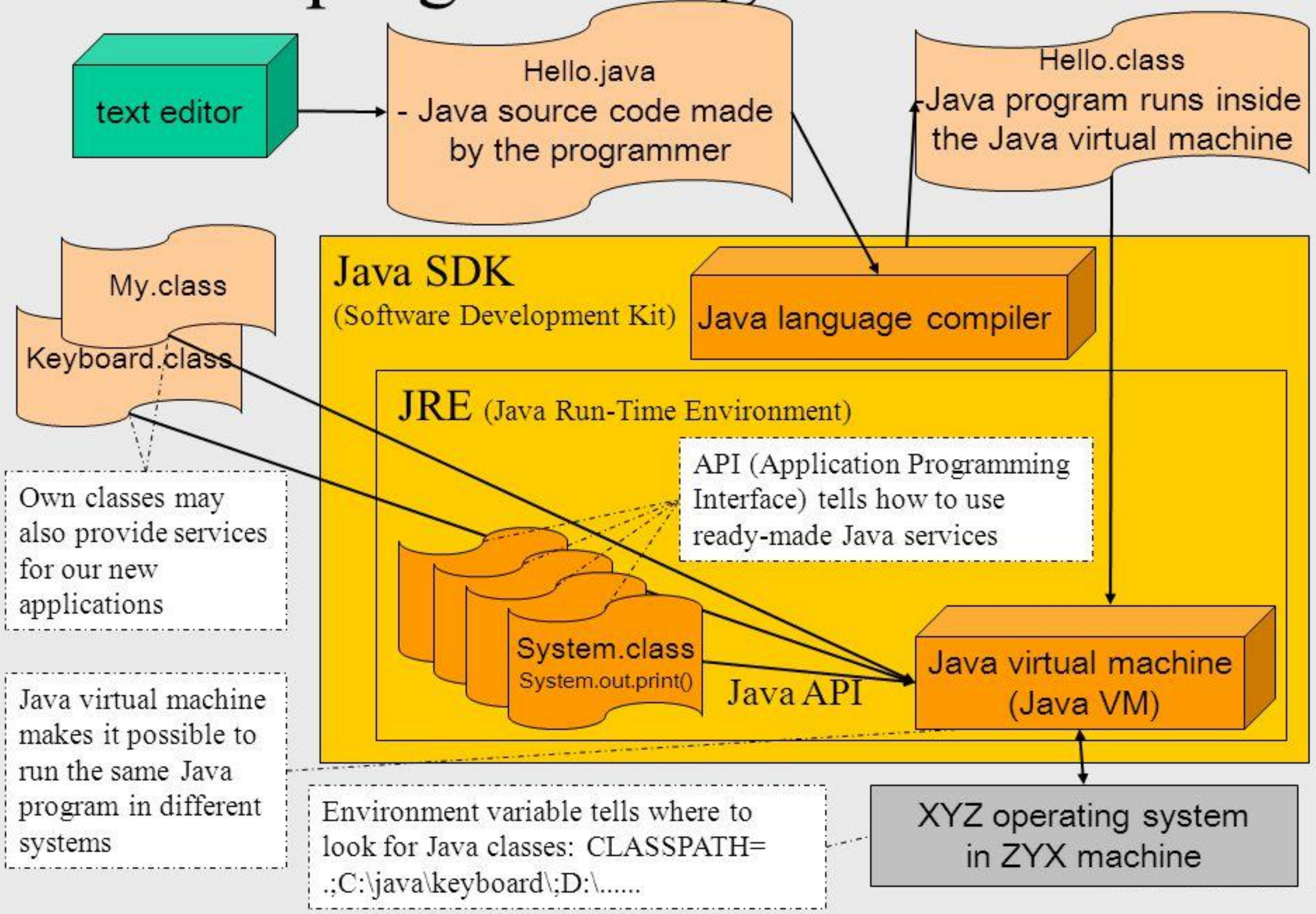
Objectives

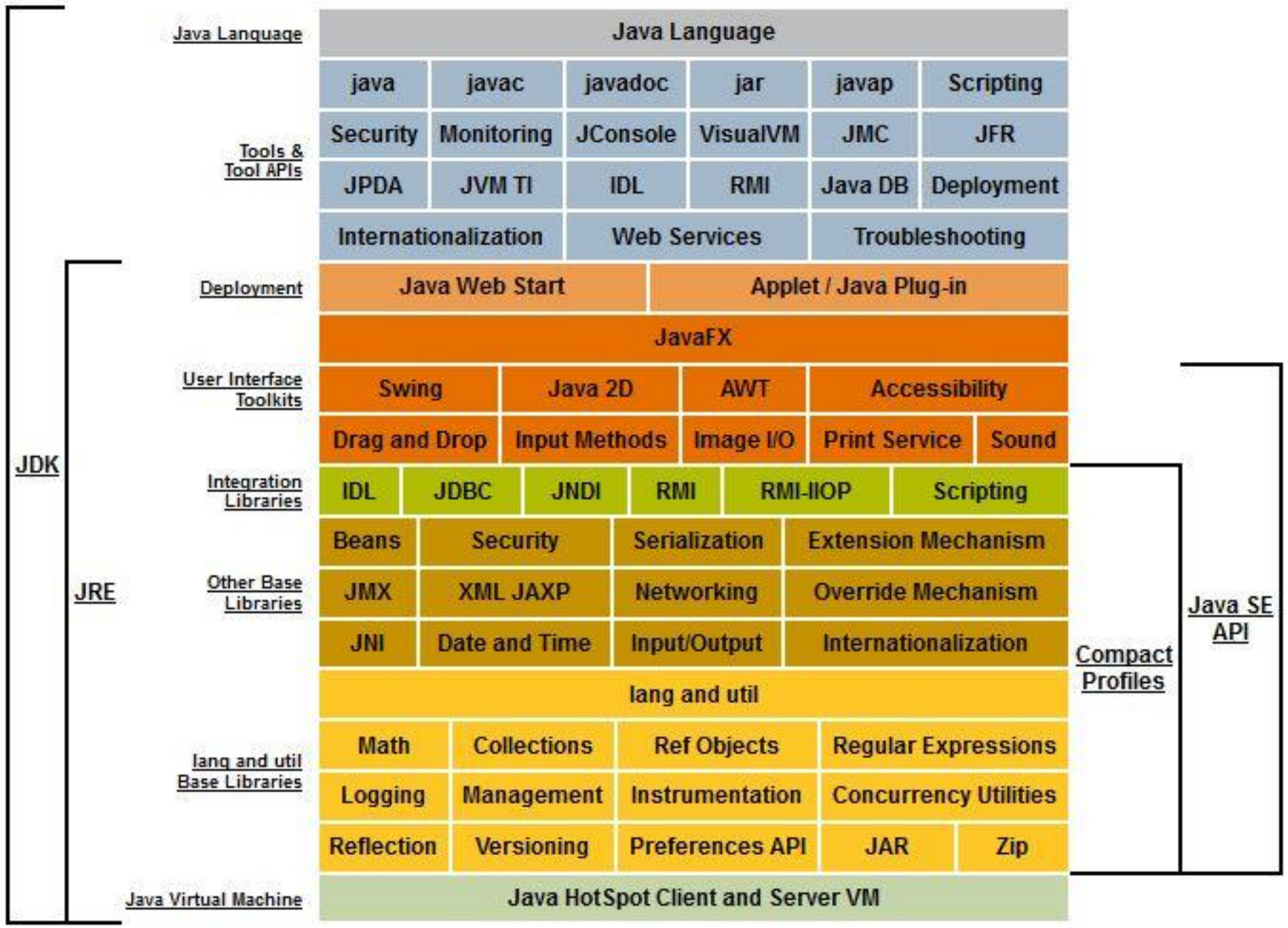
- Primitive data types beyond integers, floating point number.
- Text data types (primitive and reference data types)
- Basic I/O (System.in and System.out)
- Basic API Classes: System, Scanner, Math, Random, String, Character classes.
- Java Programming Environment.

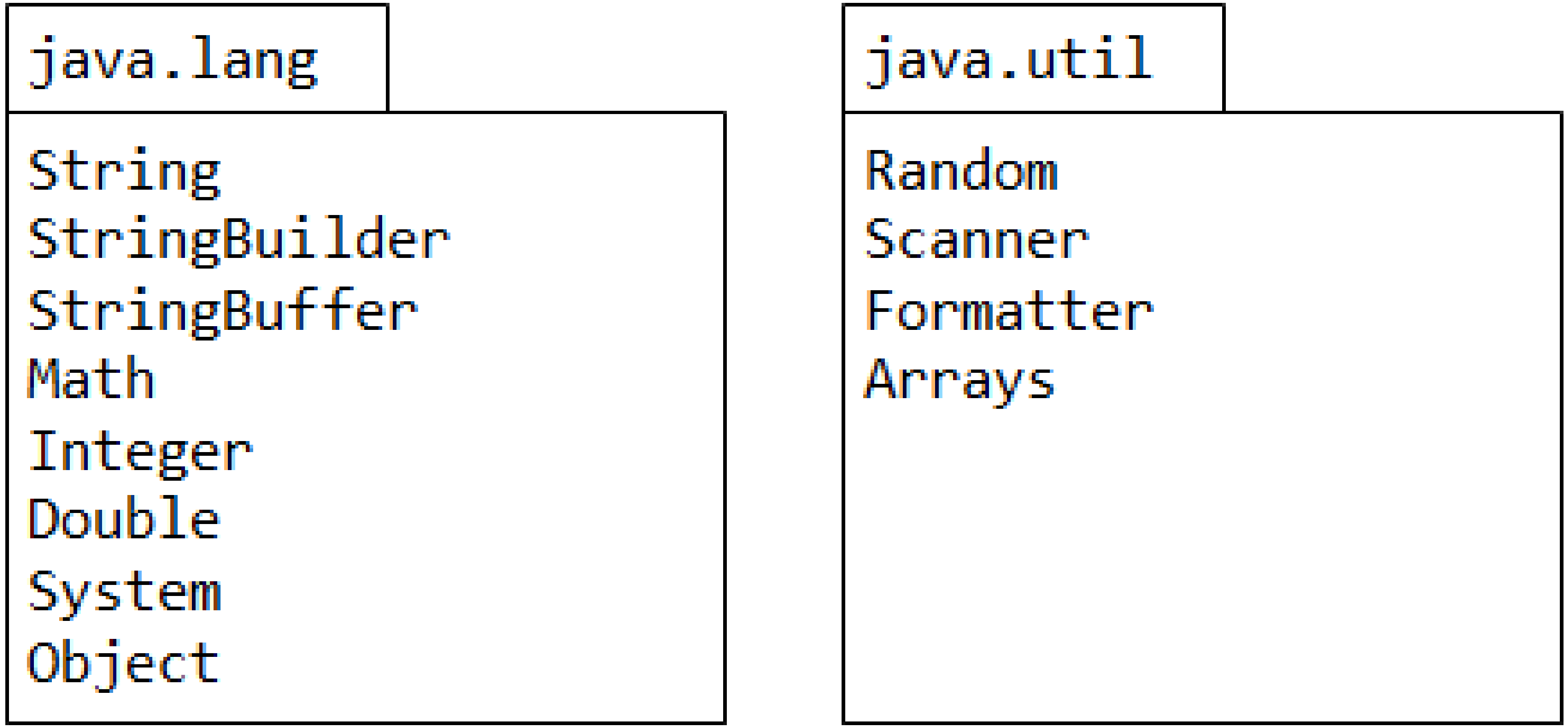
Basic Java API Overview

LECTURE 1

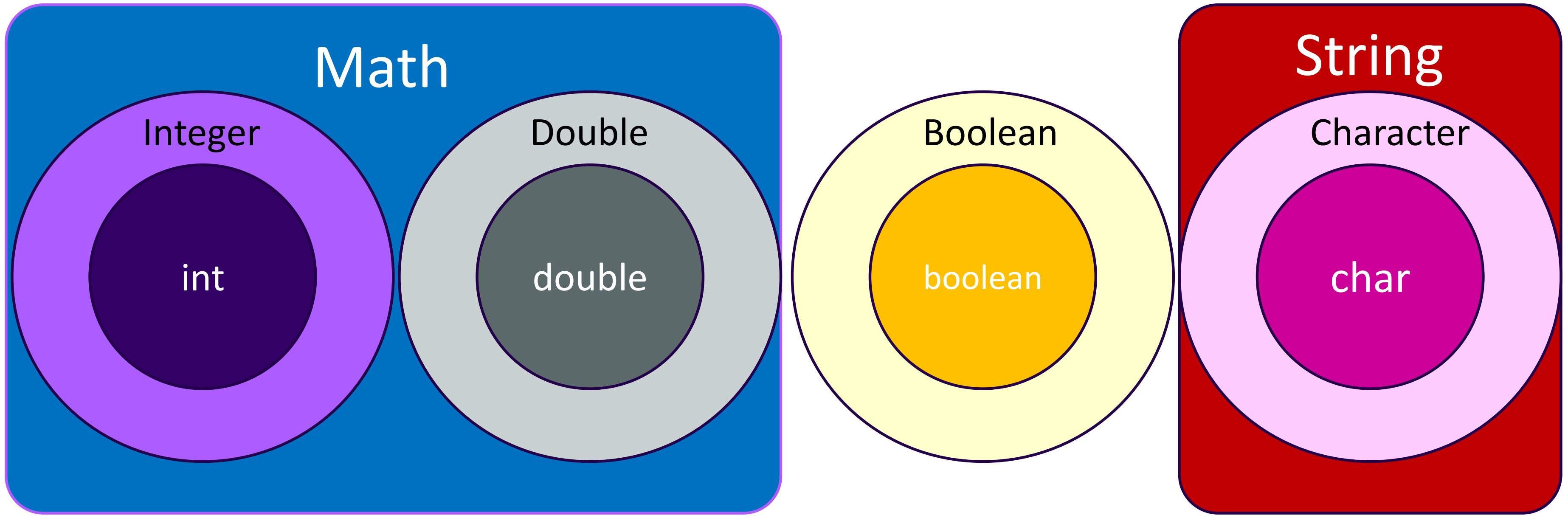
Java programming environment


text editor
Hello.java
- Java source code made by the programmer
Hello.class
-Java program runs inside the Java virtual machine
My.class
Keyboard.class
Java SDK
(Software Development Kit)
Java language compiler
JRE (Java Run-Time Environment)
API (Application Programming Interface) tells how to use ready-made Java services
System.class
System.out.print()
Java API
Java virtual machine (Java VM)
Own classes may also provide services for our new applications
Java virtual machine makes it possible to run the same Java program in different systems
Environment variable tells where to look for Java classes: CLASSPATH= .;C:\java\keyboard\;D:\......
XYZ operating system in ZYX machine


JDK
JRE
Java Language
Java Language
Tools & Tool APIs
java
javac
javadoc
jar
javap
Scripting
Security
Monitoring
JConsole
VisualVM
JMC
JFR
JPDA
JVM TI
IDL
RMI
Java DB
Deployment
Internationalization
Web Services
Troubleshooting
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Security
Serialization
Extension Mechanism
JMX
XML JAXP
Networking
Override Mechanism
JNI
Date and Time
Input/Output
Internationalization
lang and util Base Libraries
lang and util
Math
Collections
Ref Objects
Regular Expressions
Logging
Management
Instrumentation
Concurrency Utilities
Reflection
Versioning
Preferences API
JAR
Zip
Java Virtual Machine
Java HotSpot Client and Server VM
Compact Profiles
Java SE API


java.lang
String
StringBuilder
StringBuffer
Math
Integer
Double
System
Object
java.util
Random
Scanner
Formatter
Arrays

Java API Overview



System Class

LECTURE 2

The System class

- We have been using System.out.println for a while, but you might not have thought about what it means. System is a class that provides methods related to the “system” or environment where programs run. It also provides System.out, which is a special value that has additional methods (like println) for displaying output.
- In fact, we can use System.out.println to display the value of System.out:

```
System.out.println(System.out);
```

- The result is:

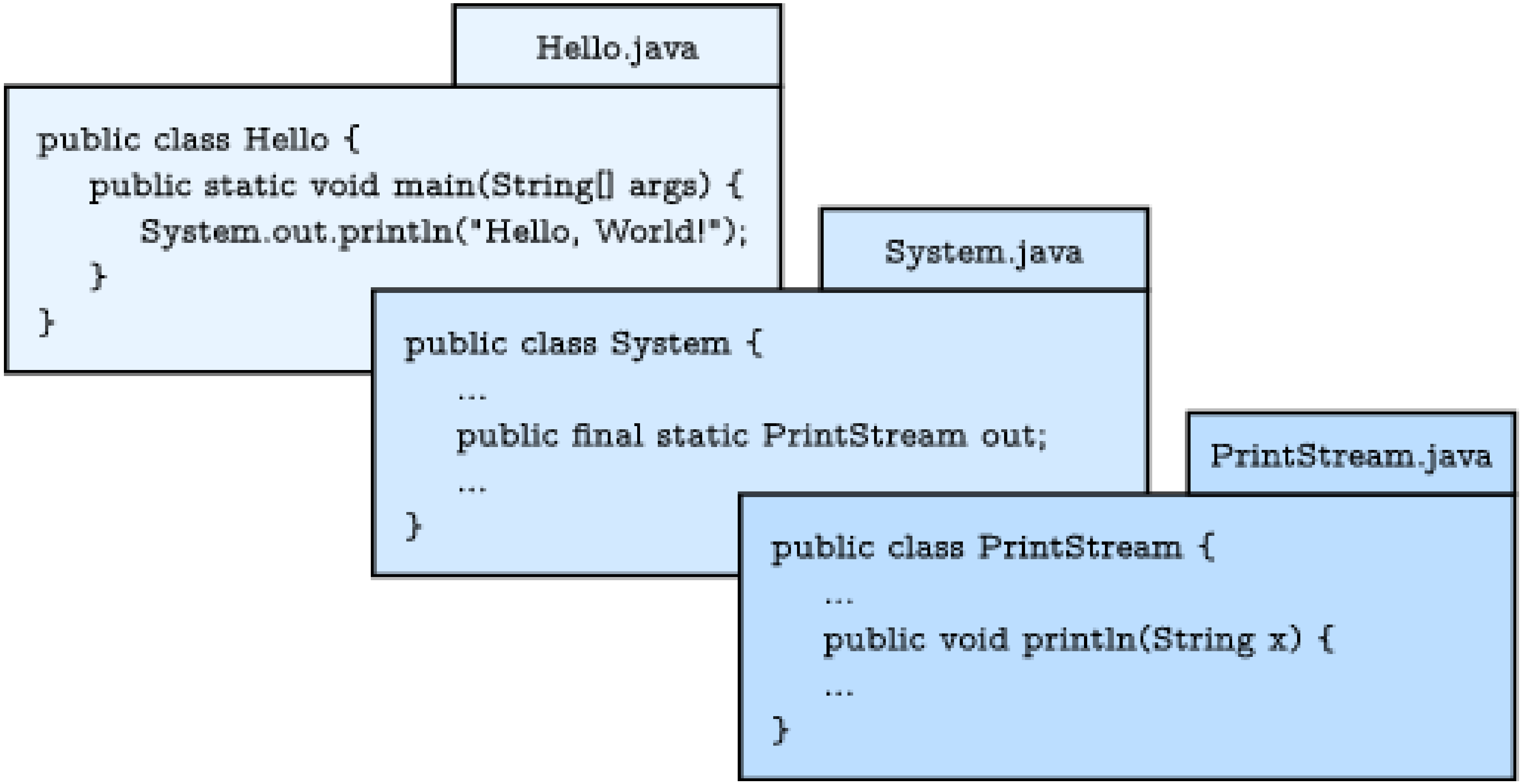
```
java.io.PrintStream@685d72cd
```

Package

- This output indicates that System.out is a **PrintStream**, which is defined in a package called java.io. A **package** is a collection of related classes; java.io contains classes for “I/O” which stands for input and output.
- The numbers and letters after the @ sign are the **address** of **System.out**, represented as a hexadecimal (base 16) number. The address of a value is its location in the computer’s memory, which might be different on different computers. In this example the address is **685d72cd**, but if you run the same code you will likely get something else.

Package

- System is defined in a file called **System.java**, and **PrintStream** is defined in PrintStream.java. These files are part of the Java **library**, which is an extensive collection of classes that you can use in your programs. The source code for these classes is usually included with the compiler.


Hello.java

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

System.java

```
public class System {
    ...
    public final static PrintStream out;
    ...
}
```

PrintStream.java

```
public class PrintStream {
    ...
    public void println(String x) {
    ...
}
```

Scanner Class

LECTURE 3

The Scanner Class

- The **System** class also provides the special value **System.in**, which is an **InputStream** that has methods for reading input from the keyboard. These methods are not convenient to use, but fortunately Java provides other classes that make it easy to handle common input tasks.
- For example, **Scanner** is a class that provides methods for inputting words, numbers, and other data. **Scanner** is provided by **java.util**, which is a package that contains various “utility classes”. Before you can use **Scanner**, you have to import it like this:

```
import java.util.Scanner;
```

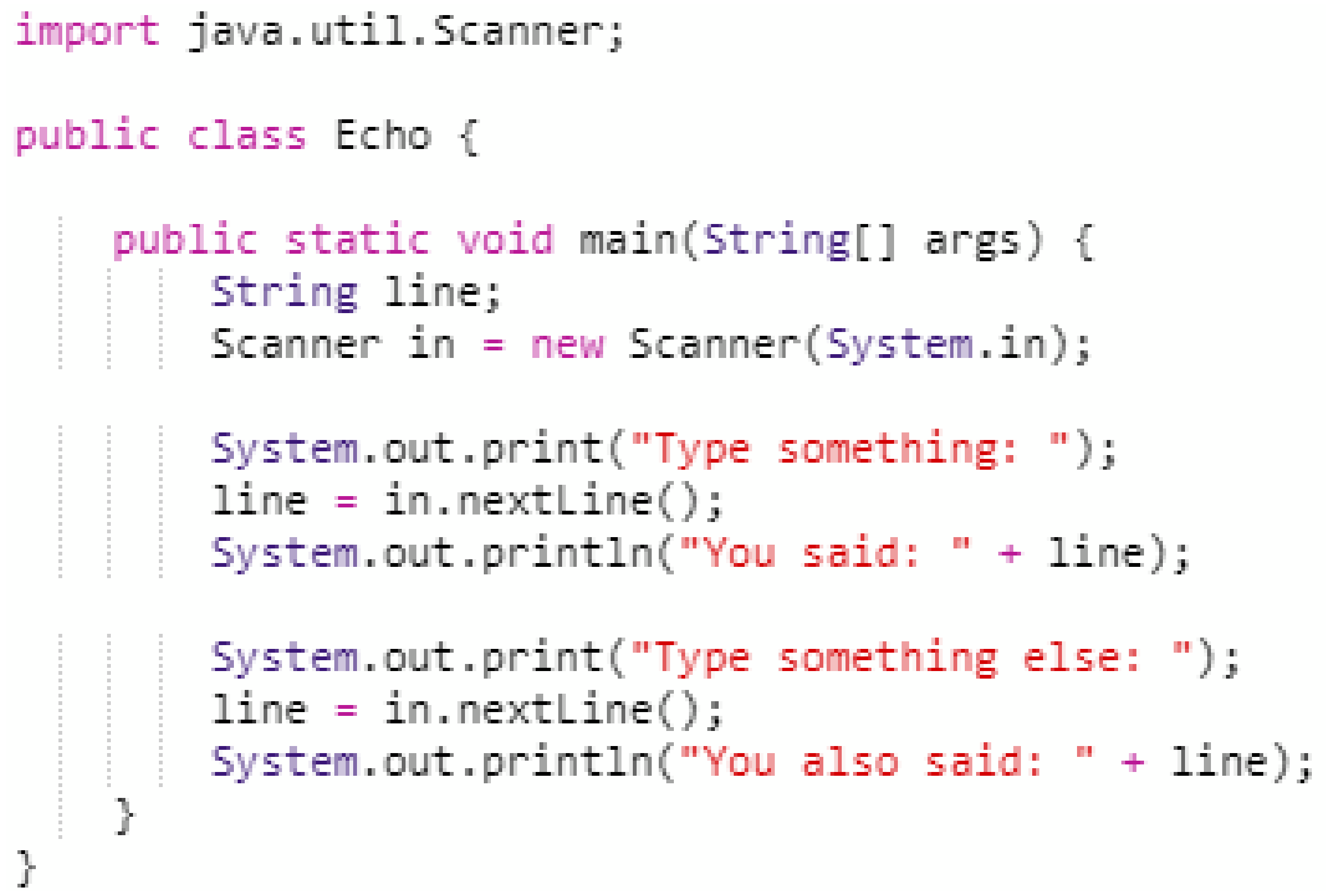
The Scanner Class

- This **import** statement tells the compiler that when you refer to Scanner, you mean the one defined in **java.util**. Using an import statement is necessary because there might be another class named Scanner in another package. Import statements can’t be inside a class definition. By convention, they are usually at the beginning of the file.
- Next you have to initialize the **Scanner**. This line declares a Scanner variable named in and creates a Scanner that reads input from System.in:

```
Scanner in = new Scanner(System.in);
```

The Scanner Class

The Scanner class provides a method called **nextLine** that reads a line of input from the keyboard and returns a **String**. The following example reads two lines and repeats them back to the user:



```java
import java.util.Scanner;

public class Echo {

    public static void main(String[] args) {
        String line;
        Scanner in = new Scanner(System.in);

        System.out.print("Type something: ");
        line = in.nextLine();
        System.out.println("You said: " + line);

        System.out.print("Type something else: ");
        line = in.nextLine();
        System.out.println("You also said: " + line);
    }
}
```

Demonstration Program

ECHO.JAVA

The Scanner Class

- If you omit the import statement at the top of the file, you will get a compiler error saying “cannot find symbol”. That means the compiler doesn’t know where to find the definition for **`Scanner`**.
- You might wonder why we can use the **`System`** class without importing it. **`System`** belongs to the `java.lang` package, which is imported automatically.
- According to the documentation, **`java.lang`** “provides classes that are fundamental to the design of the Java programming language.” The `String` class is also part of the **`java.lang`** package

Boolean Data Type

LECTURE 4

Java boolean values boolean constants

- Only two boolean value: **true**, **false**.
- They are also called 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;
```

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

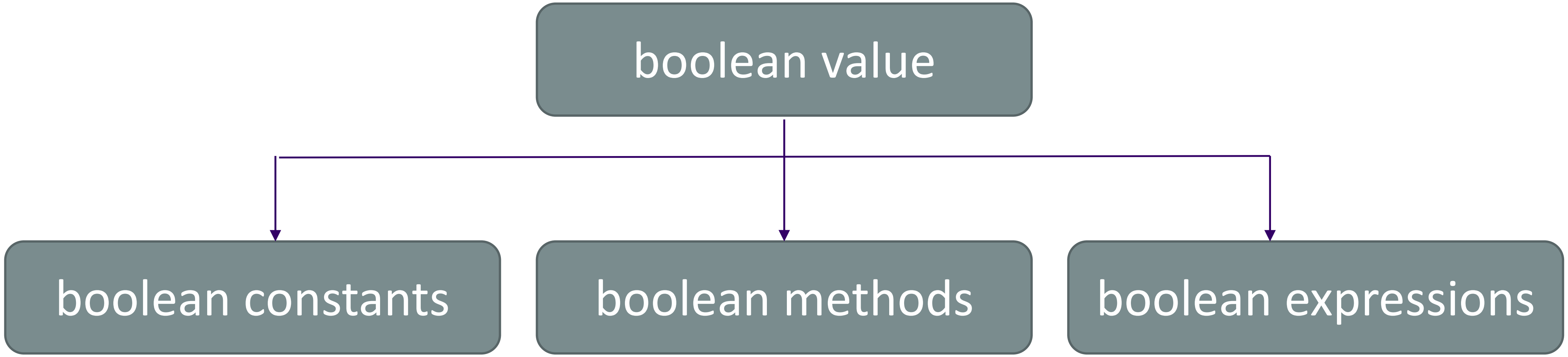
For example, suppose you have declared two variables: **int i = 5;** **int j = 10;**

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 function (methods)

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

boolean values



true
final boolean YES = **true**;

isLetter()
isLowerCase()

a >= 3

Logic Operators

LECTURE 5

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)

A	B	A | B	A & B	A ^ B	!A
False	False	False	False	False	True
True	False	True	False	True	False
False	True	True	False	True	True
True	True	True	True	False	False

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
false	false	false
false	true	true
true	false	true
true	true	true

Example (assume age = 24, gender = 'F')

(age > 34) || (gender == 'F') is true, because (gender == 'F') is true.

(age > 34) || (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false.

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);
    }
```

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.

Character Data Type

LECTURE 6

Character Data Type (char)

- A character data type (1 byte = 8 bits) represents a single character.
- For Java, it is 16bits (2 bytes) to accommodate Unicode. (UTF-16)

```
char letter = 'A';                (ASCII)
char numChar = '4';               (ASCII)
char letter = '\u0041';           (Unicode)
char numChar = '\u0034';             (Unicode)
```

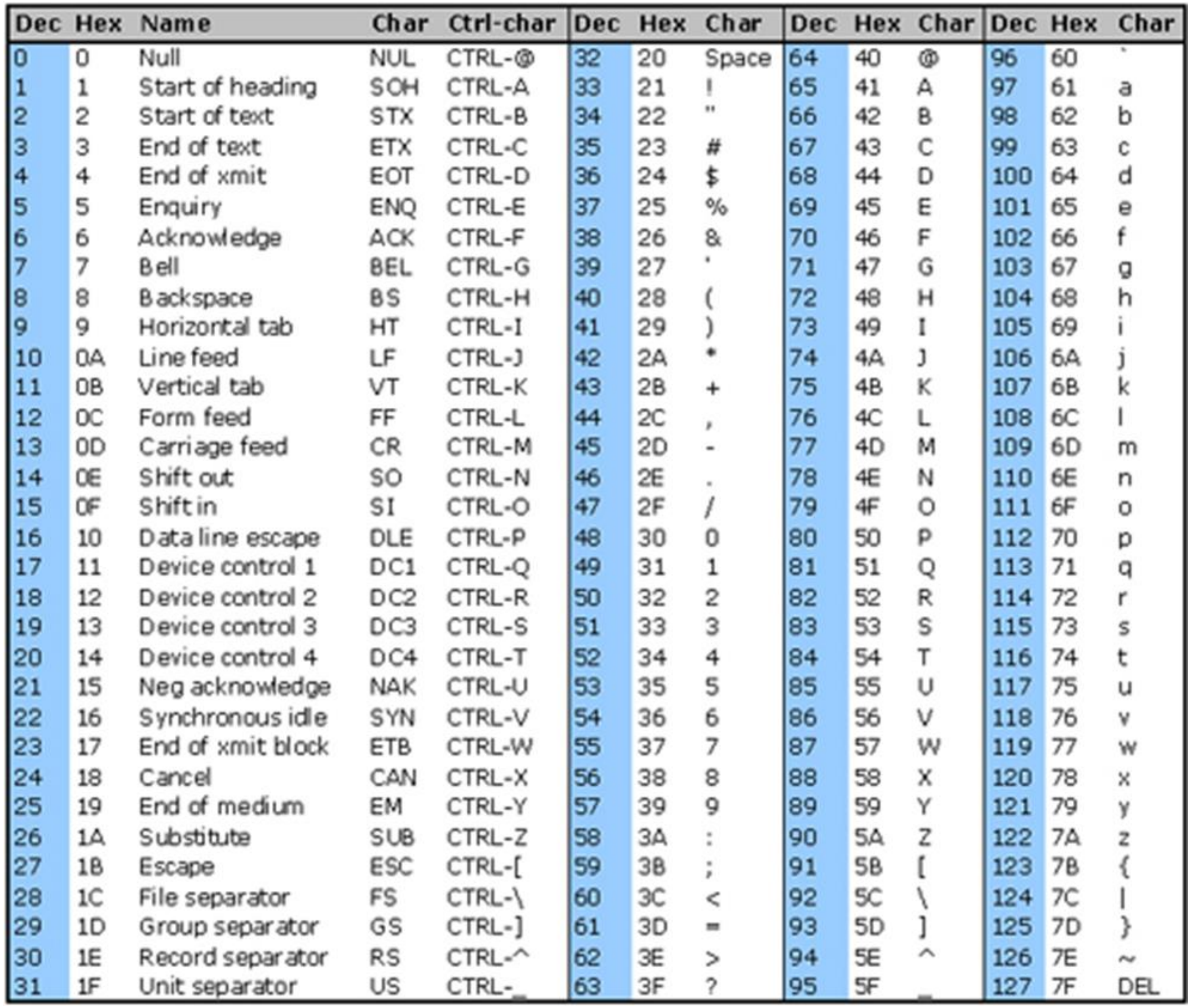
- char is one primitive data type.
- char is a data type for symbols.
- char is also an unsigned integer type

String Literal

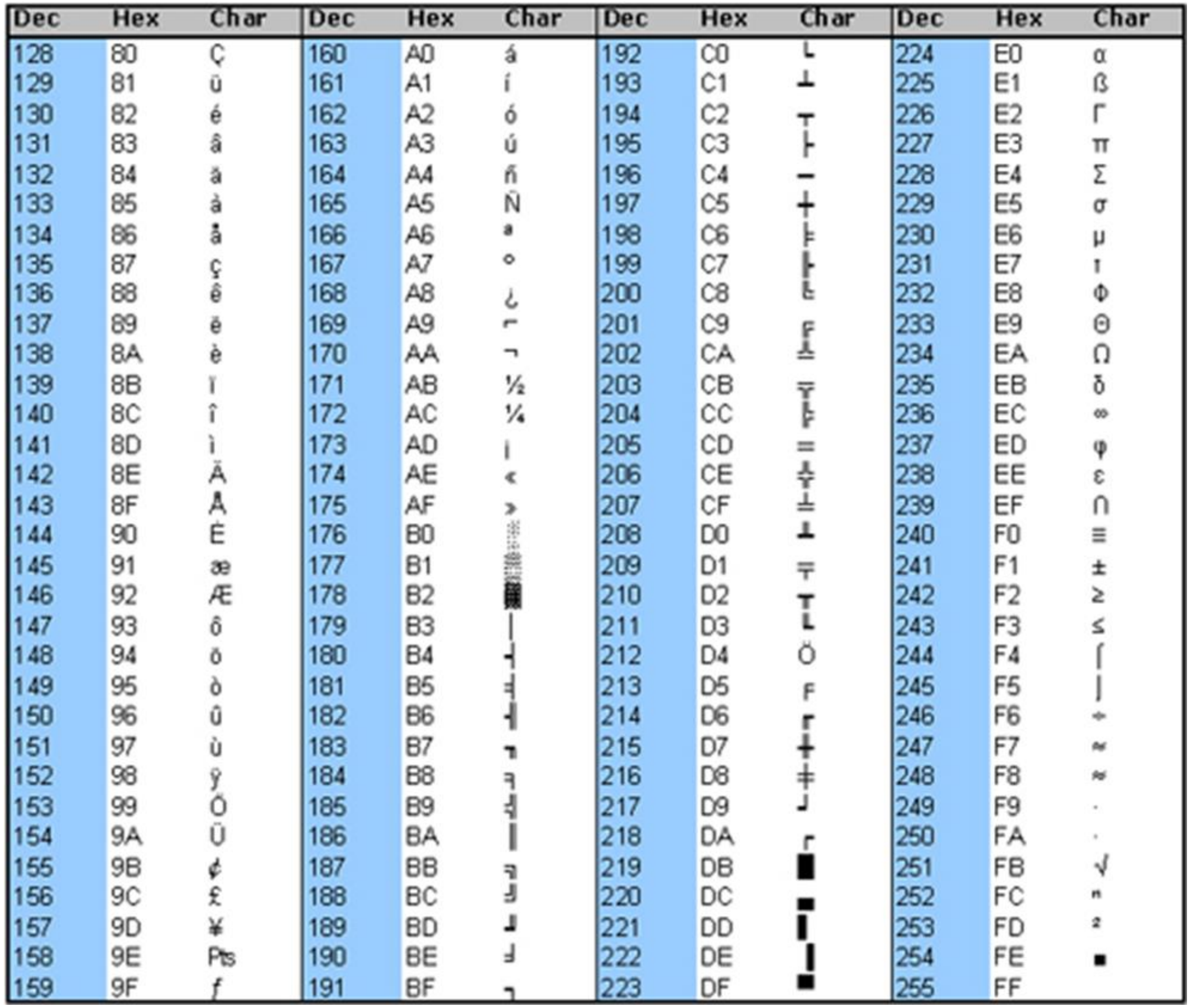
- A string literal must be enclosed in quotation marks (“ “). A character literal is a single character enclosed in a single quotations marks (‘ ‘). Therefore, **“A” is a string**, but **‘A’ is a character**.

- **NOTE:** The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

```
char ch = 'a';
System.out.println(++ch);
```



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

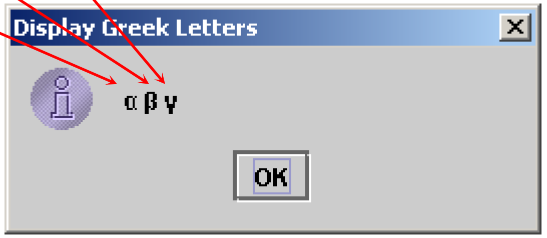


Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	Ö	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	≈
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	·
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Unicode Format

Java characters use ***Unicode***, a 16-bit encoding scheme established by the Unicode Consortium to support the interchange, processing, and display of written texts in the world's diverse languages. Unicode takes two bytes, preceded by \u, expressed in four hexadecimal numbers that run from <u>'\u0000'</u> to <u>'\uFFFF'</u>. So, Unicode can represent `65535 + 1 characters.`

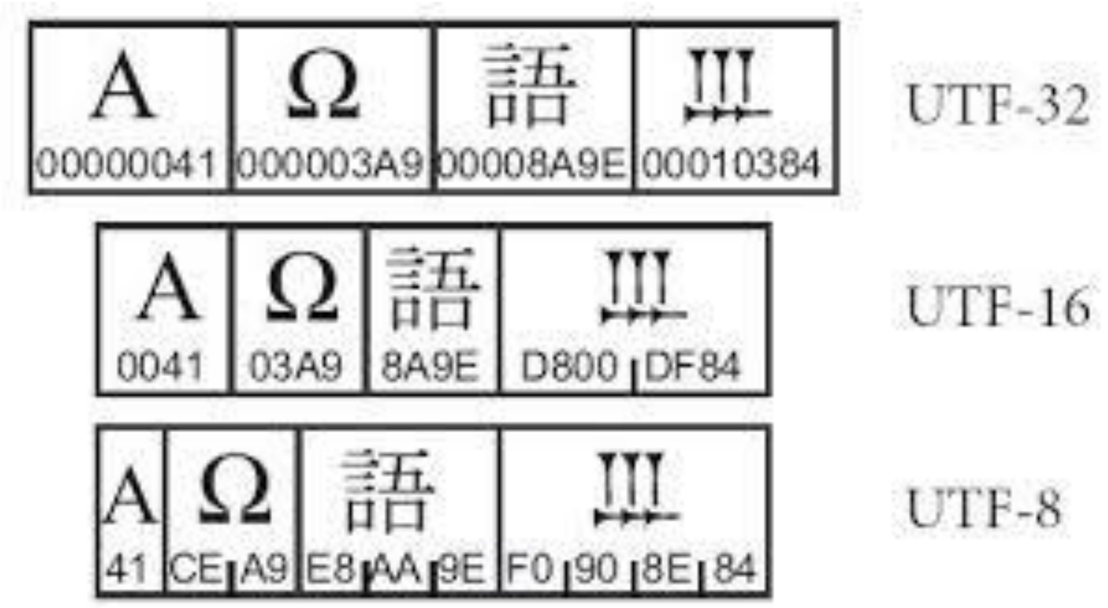
Unicode \u03b1 \u03b2 \u03b3 for three Greek letters



Unicode

- International standard for representing written language in computers
- Latest version 5.2 adds 6648 new characters including support for Vedic Sanskrit
- Maintained in sync with ISO 10646
- Three main encodings: UTF-8, UTF-16 and UTF-32
- Address space of 21 bits

ASCII Code, Unicode, UTF-8, UTF-16, UTF-32



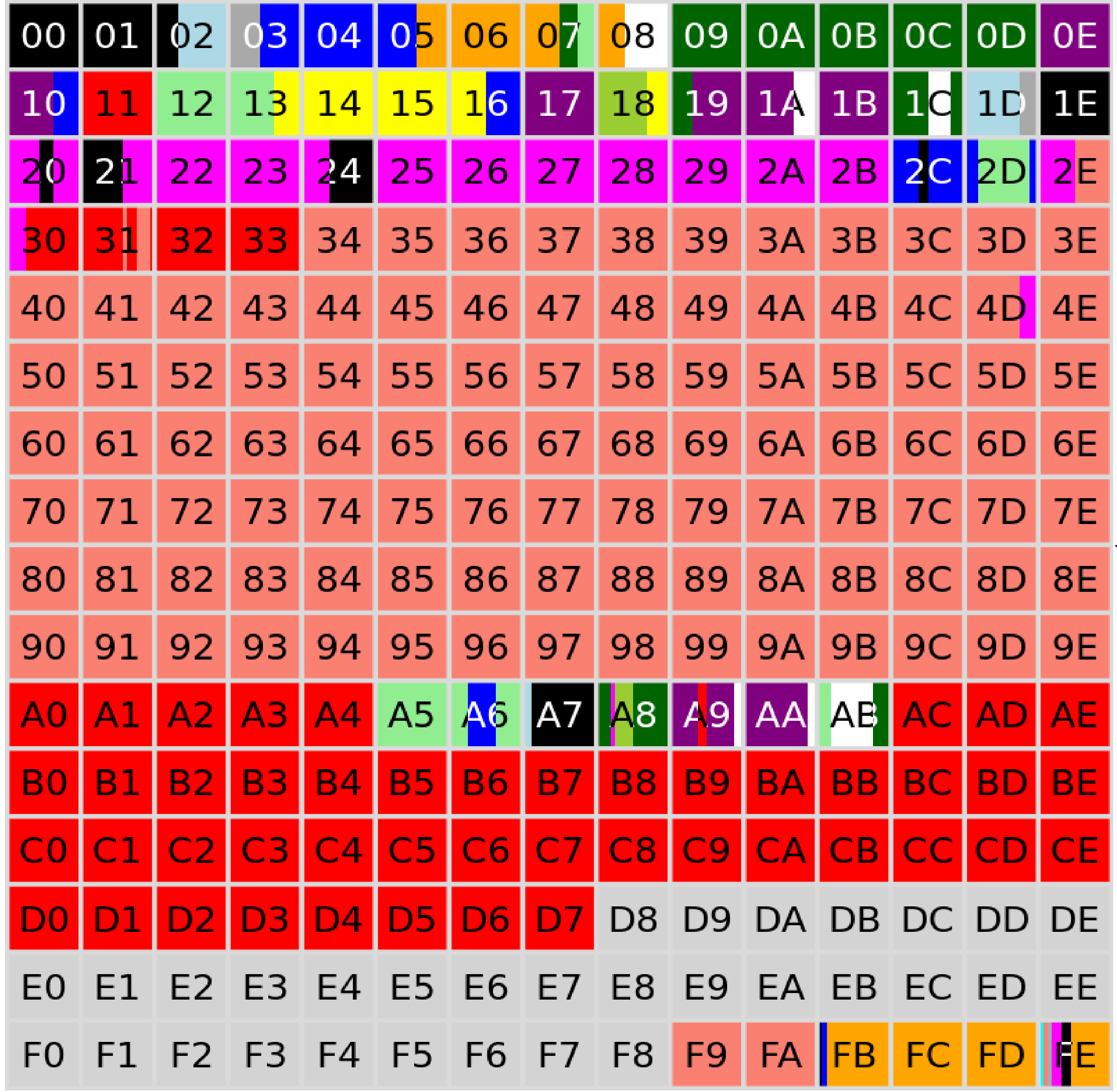
ASCII

UTF-16

Types of Characters	First 16 Bits	Second 16 Bits
ASCII	0000-007F	-
European (Except ASCII), Arabic, Hebrew	0080-07FF	-
Iindic, Thai, certain symbols (such as the euro symbol), Chinese, Japanese, Korean	0800-0FFF 1000 - CFFF D000 - D7FF F900 - FFFF	-
Private Use Area #1	E000 - EFFF F000 - F8FF	-
Supplementary characters: Additional Chinese, Japanese, and Korean characters; historic characters; musical symbols; mathematical symbols	D800 - D8BF D8CO - DABF DAC0 - DB7F	DC00 - DFFF DC00 - DFFF DC00 - DFFF
Private Use Area #2	DB80 - DBBF DBC0 - DBFF	DC00 - DFFF DC00 - DFFF

UTF-16

Types of Characters	First Byte	Second Byte	Third Byte	Fourth Byte
ASCII	00 - 7F	-	-	-
European (except ASCII), Arabic, Hebrew	C2 - DF	80 - BF	-	-
Indic, Thai, certain symbols (such as the euro symbol), Chinese, Japanese, Korean	E0 E1 - EC ED EF	A0 - BF 80 - BF 80 - 9F A4 - BF	80 - BF 80 - BF 80 - BF 80 - BF	-
Private Use Area #1	EE EF	80 - BF 80 - A3	80 - BF 80 - BF	-
Supplementary characters: Additional Chinese, Japanese, and Korean characters; historic characters; musical symbols; mathematical symbols	F0 F1 - F2 F3	90 - BF 80 - BF 80 - AF	80 - BF 80 - BF 80 - BF	80 - BF 80 - BF 80 - BF
Private Use Area #2	F3 F4	B0 - BF 80 - 8F	80 - BF 80 - BF	80 - BF 80 - BF



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE

Unicode
Basic
Multilingual
Plane
1st 2Hexa-digits

\u2EXX to
\u9FXX

Almost for
Chinese
(East Asian CJKV)

The `Character` Class (Wrapper Class)

java.lang.Character	
+Character(value: char)	Constructs a character object with char value
+charValue(): char	Returns the char value from this object
+compareTo(anotherCharacter: Character): int	Compares this character with another
+equals(anotherCharacter: Character): boolean	Returns true if this character equals to another
+isDigit(ch: char): boolean	Returns true if the specified character is a digit
+isLetter(ch: char): boolean	Returns true if the specified character is a letter
+isLetterOrDigit(ch: char): boolean	Returns true if the character is a letter or a digit
+isLowerCase(ch: char): boolean	Returns true if the character is a lowercase letter
+isUpperCase(ch: char): boolean	Returns true if the character is an uppercase letter
+toLowerCase(ch: char): char	Returns the lowercase of the specified character
+toUpperCase(ch: char): char	Returns the uppercase of the specified character

Examples

```
Character charObject = new Character('b');

charObject.compareTo(new Character('a')) returns 1
charObject.compareTo(new Character('b')) returns 0
charObject.compareTo(new Character('c')) returns -1
charObject.compareTo(new Character('d') returns –2
charObject.equals(new Character('b')) returns true
charObject.equals(new Character('d')) returns false
```

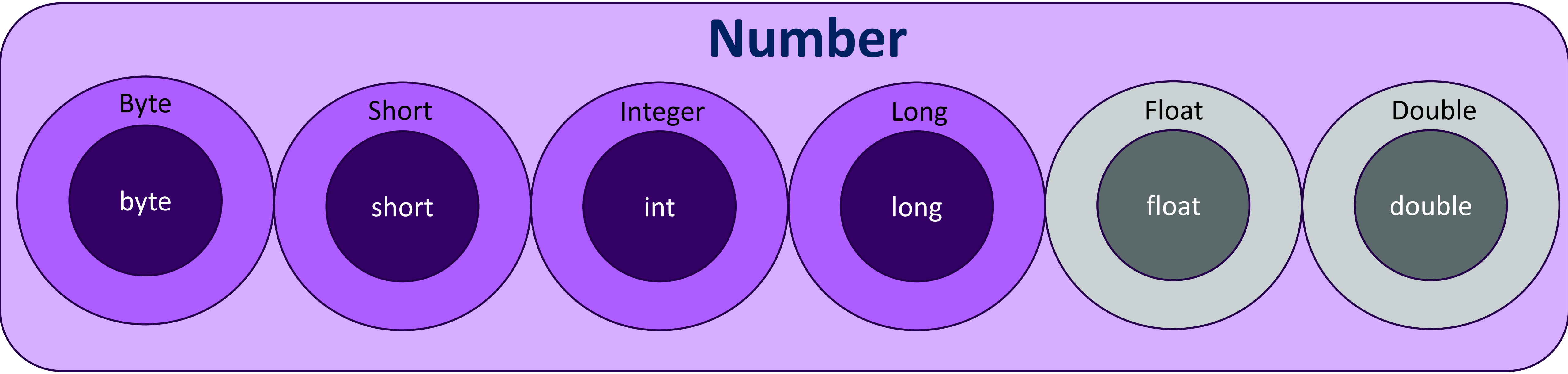
Demonstration Program

YESNOHAPPY.JAVA

Math/Number Classes

LECTURE 7

Number Classes Overview



Math Class

- The **java.lang.Math** class contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

Mathematical Methods

- The mathematical Functions are methods in the class Math. Math class is a object class with many methods. The object itself does not seem to represent much of the meaning. We can view this Math class as a library of functions instead.
- The methods can be categorized as trigonometric methods, exponent methods, and service methods.
- Service methods including rounding, minimum, maximum, absolute, and random methods.

Mathematical Constants

- In addition to methods, the Math class provides tow useful double constants, **PI** and **E** (the base of natural logarithms. You can use these constants as **Math.PI** and **Math.E** in any program.

Math.PI = 3.141592…

Math.E = 2.718…

Integer.MIN_VALUE = -2147483648

Integer.MAX_VALUE = 2147483647

Double.MIN_VALUE = 4.9E-324

Double.MAX_VALUE = 1.7976931348623157E308

Double.POSITIVE_INFINITY (∞)

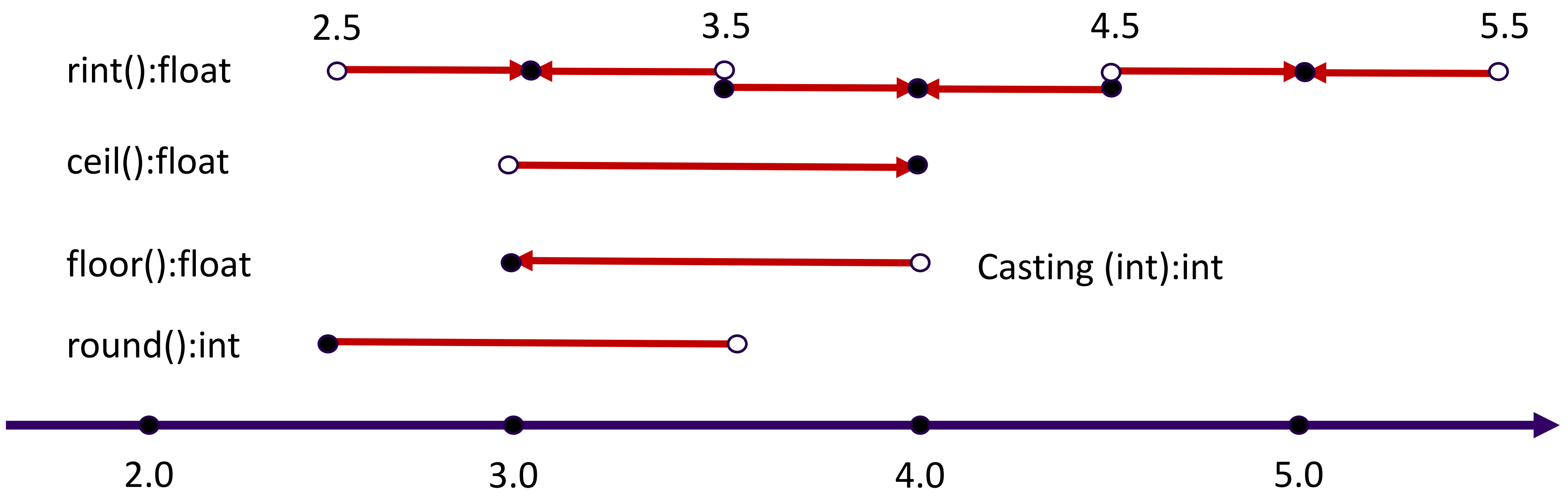
Double.NEGATIVE_INFINITY (-∞)

Double.NaN (Not a Number)

The Service Methods

Rounding Methods	Description
ceil(x)	x is rounded up to its nearest integer. This integer is returned as a double value.
floor(x)	x is rounded down to its nearest integer. This integer is returned as a double value.
rint(x)	x is rounded up to its nearest integer. If x is equally close to two integers, the even one is returned as a double value
round(x)	Returns (int) Math.floor(x+0.5) if x is a float and returns (long) Math.floor(x+0.5) if x is a double.
Min Max Abs Methods	**Description**
max(x, y)	Return the greater number between x and y.
min(x, y)	Return the less number between x and y
abs(x)	Return the absolute value of x
Random Methods	**Description**
random()	Return a random number between $0 \leq y < 1$. (Compared to java.util.Random)

Rounding Methods



Math.Random

Usage:

```
final int ELEMENT_COUNT  = 10;
final int STEPS = 2;
final int BASELINE = 11;
int randNum = (int) (Math.random()*ELEMENT_COUNT) * STEPS + BASELINE;
```

(13-11) = 2

This generates a random number in SET = [11, 13, 15, 17, 19, 21, 23, 25, 27, 29]

The Exponent Methods

Methods	Description	
exp(x)	Returns e raised to power of x	(e^x)
log(x)	Return the natural logarithm of x	(ln(x) = $\log_e$(x)).
log10(x)	Returns the base 10 algorithm of x	($\log_{10}$(x)).
pow(a, b)	Returns a raised to the power of b	(a^b)
sqrt(x)	Returns the square root of x	$(\sqrt{x})$ for x >= 0

Trigonometric Methods

Methods	Description
sin(radians)	Returns the trigonometric sine of an angle in radians
cos(radians)	Returns the trigonometric cosine of an angle in radians
tan(radians)	Returns the trigonometric tan of an angle in radians
toRadians(degree)	Return the angle in radians for the angle in degree.
toDegree(radians)	Return the angle in degrees for the angle in radians.
asin(a)	Return the angle in radians for the inverse of sine.
acos(a)	Return the angle in radians for the inverse of cosine.
atan(a)	Return the angle in radians for the inverse of tangent.

Case Study Computing Angles of Triangle

Cosine Rule:

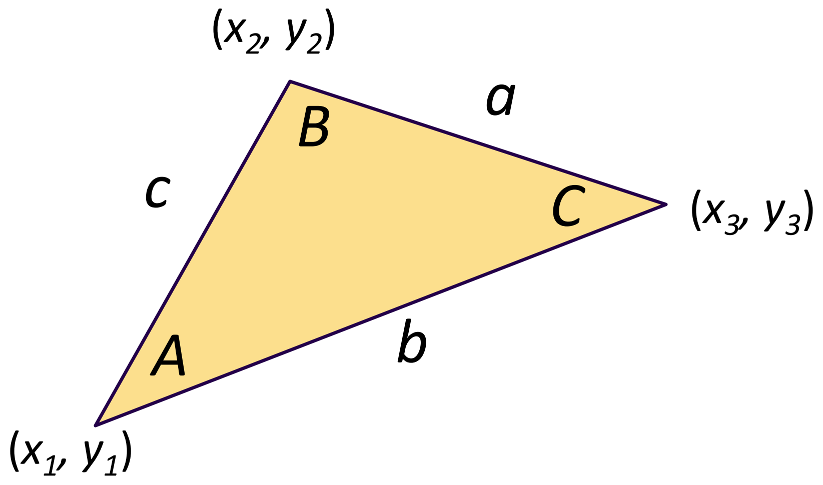
$$a^2 = b^2 + c^2 - 2\ b\ c \cos A$$

To find an angle:

$$A = \cos^{-1}\left(\frac{a^2 - b^2 - c^2}{-2bc}\right) = \text{acos}\left(\frac{a^2 - b^2 - c^2}{-2bc}\right)$$

$$B = \cos^{-1}\left(\frac{b^2 - a^2 - c^2}{-2ac}\right) = \text{acos}\left(\frac{b^2 - a^2 - c^2}{-2ac}\right)$$

$$C = \cos^{-1}\left(\frac{c^2 - b^2 - a^2}{-2ab}\right) = \text{acos}\left(\frac{c^2 - b^2 - a^2}{-2ab}\right)$$



Try ComputeAngles.java

Demonstration Program

COMPUTEANGLE.JAVA

Random Class (Non-AP)

LECTURE 8

Random class Example

```
import java.util.Random;
/** Generate 10 random integers in the range 0..99. */
public final class RandomInteger {
  public static final void main(String... aArgs){
    log("Generating 10 random integers in range 0..99.");
    //note a single Random object is reused here
    Random randomGenerator = new Random();
    for (int idx = 1; idx <= 10; ++idx){
      int randomInt = randomGenerator.nextInt(100);
      log("Generated : " + randomInt);
    }
    log("Done.");
  }
  private static void log(String aMessage){
    System.out.println(aMessage);
  }
}
```

Random class Example

Example run of this class:

```
Generating 10 random integers in range
0..99.
Generated : 44
Generated : 81
Generated : 69
Generated : 31
Generated : 10
Generated : 64
Generated : 74
Generated : 57
Generated : 56
Generated : 93
Done.
```

Demonstration Program

RANDOMINTEGER.JAVA

Random Class

- Create a Random input stream:

```
int seed = 1125;
Random randGen = new Random(seed);
```

- Get a integer:

```
int a = randGen.nextInt(100);
```

// 100 is ELEMENT_COUNT equivalent in our previous lecture.

- Proper **seed** needed.

Math.random()

- No constructor needed.
- No integer return. Only double return.
- Randomness by System clock. No need to input proper seed.

Demonstration Program

RANDOMDEMO.JAVA

Lab

SUBTRACTION QUIZ

Lab: SubtractionQuiz.java

- Write a program to generate two random integers, calculate the difference between the two numbers. Then, ask user to input the answer.
- If the input value equals the difference by calculation, then print out a message to notify user the answer is correct.
- Otherwise, print out a message to notify the user that his answer is wrong and provide the correct answer for the difference.

Lab: Work on your own.

1. Download SubtractionQuiz.java to work on your own.

2. After you finish this lab, download SubtractionQuizAnswer.java for your own reference.

Good Luck !!!

Project: SubtractionQuiz.java

Student should work on this project in Class.

Demonstration Program

SCANNERRANDOM.JAVA

Basic Boolean Algebra

LECTURE 9

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

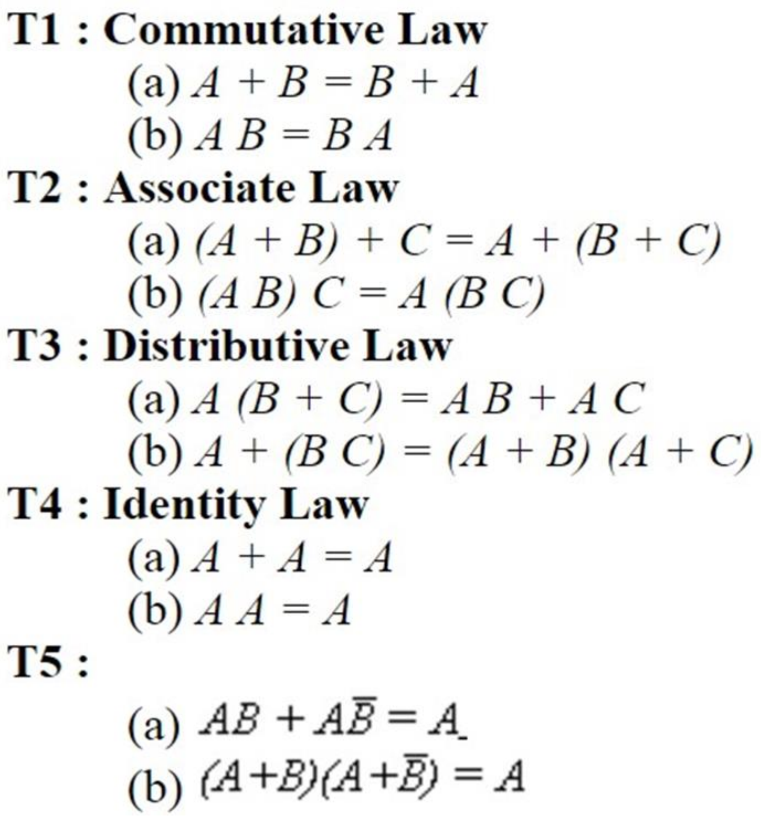
•Rule 7: 1 + 0 = 0 + 1 = 1

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

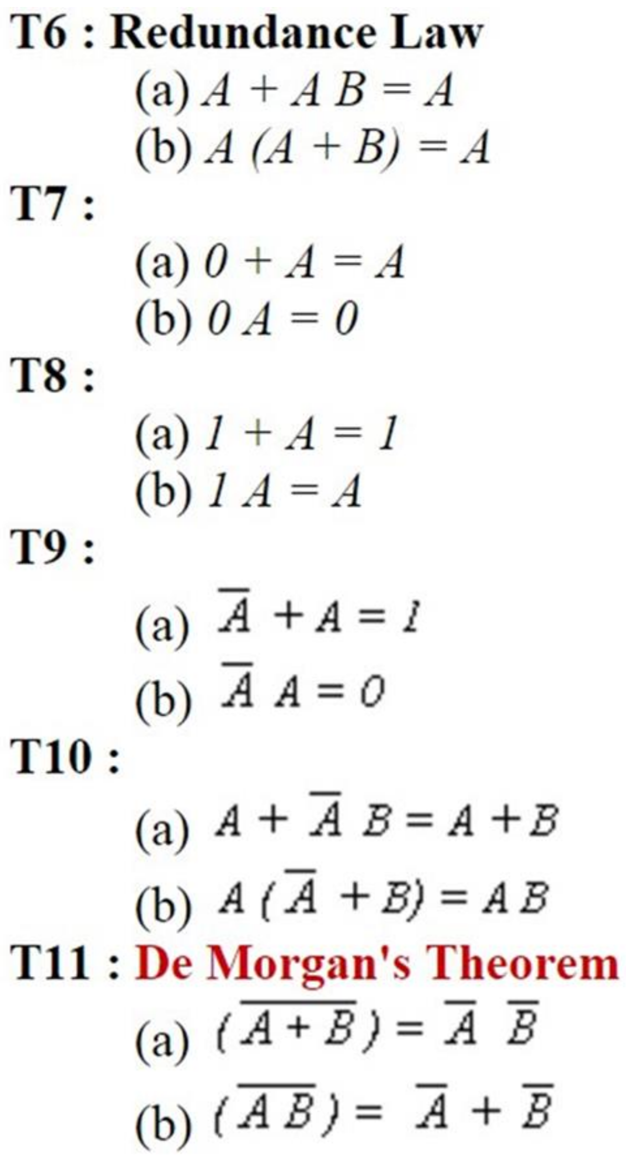
(a) $A + A = A$

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

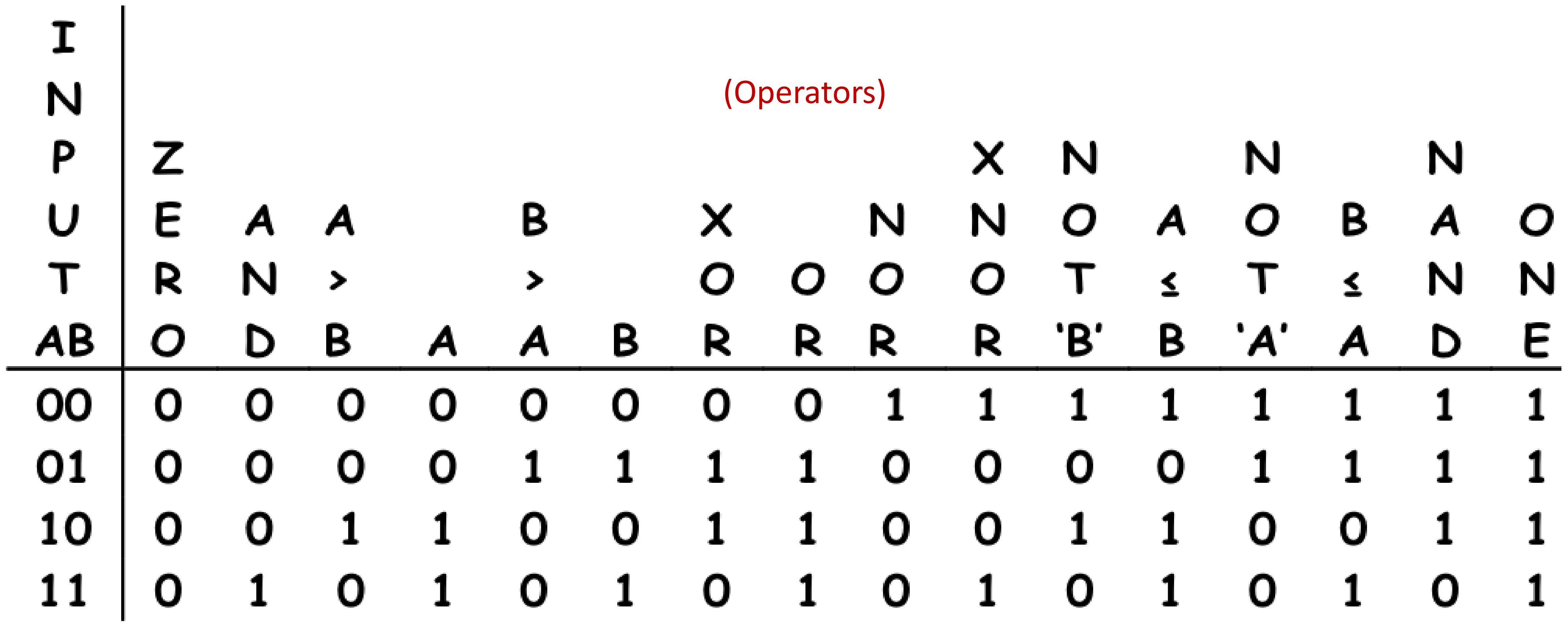
T11 : De Morgan's Theorem

(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)

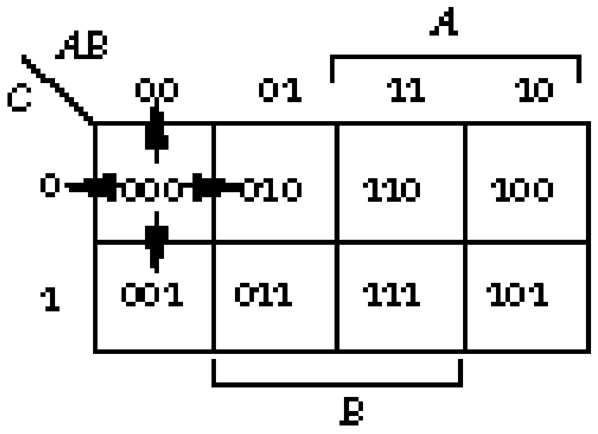


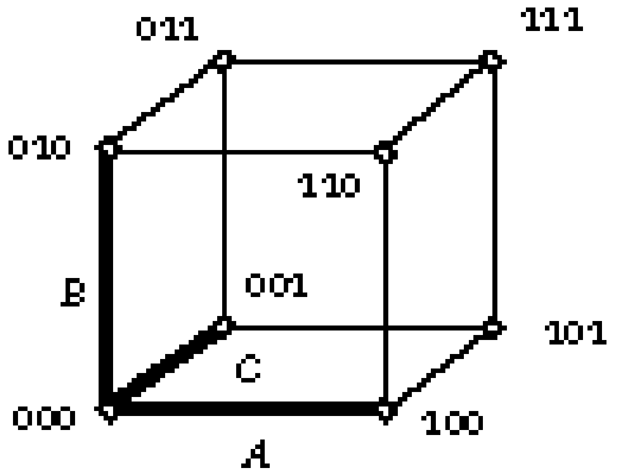
(Operators)

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 10

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)

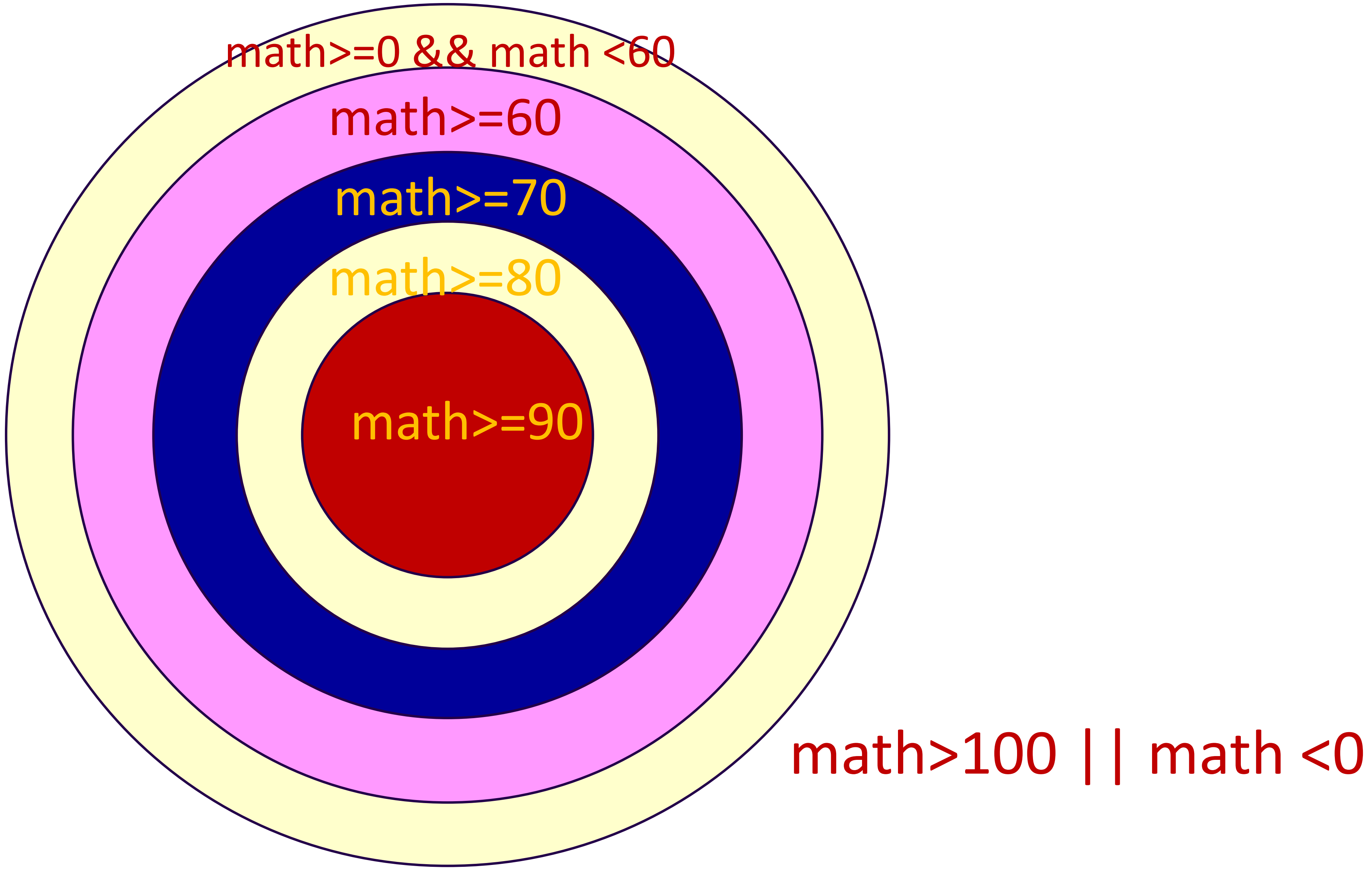
Smaller set go first:

```
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



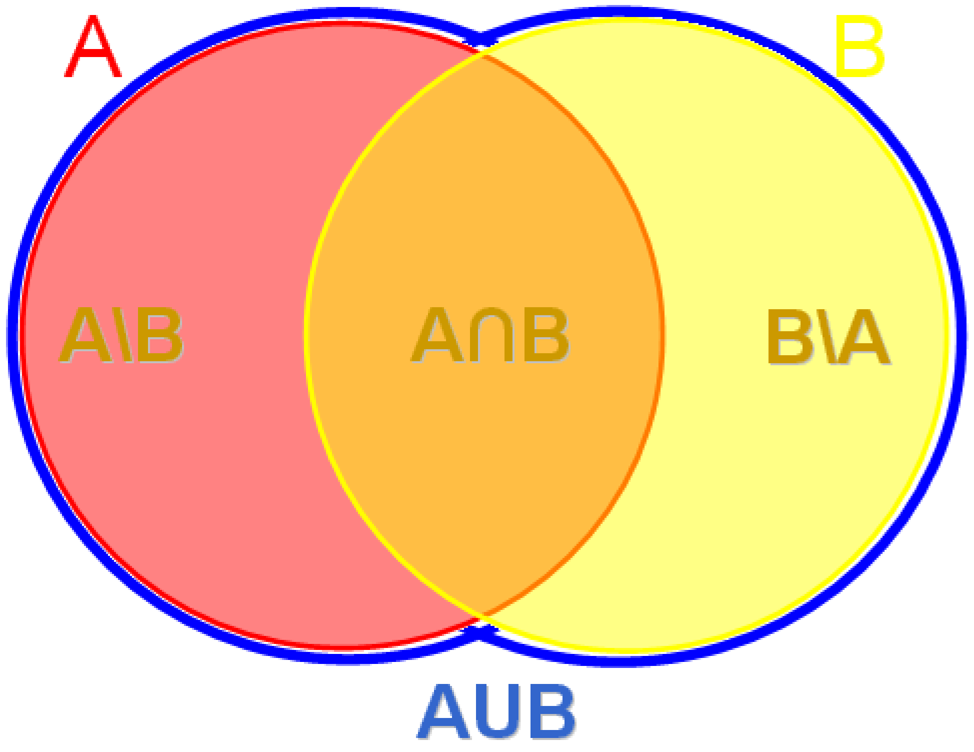
Set Theory and Logic Design (only positive numbers are shown)

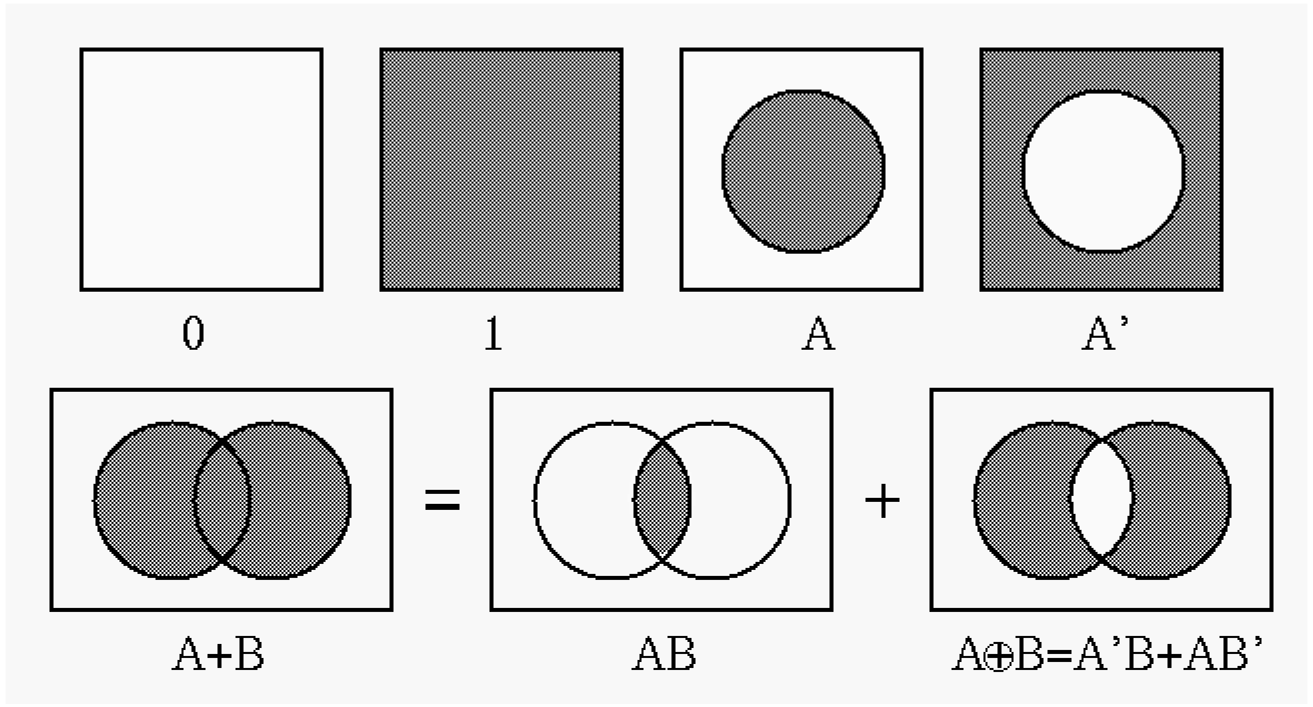
boolean m5 = (x % 5 == 0); // Set S5 = [0, 5, 10, 15, 20, ...]

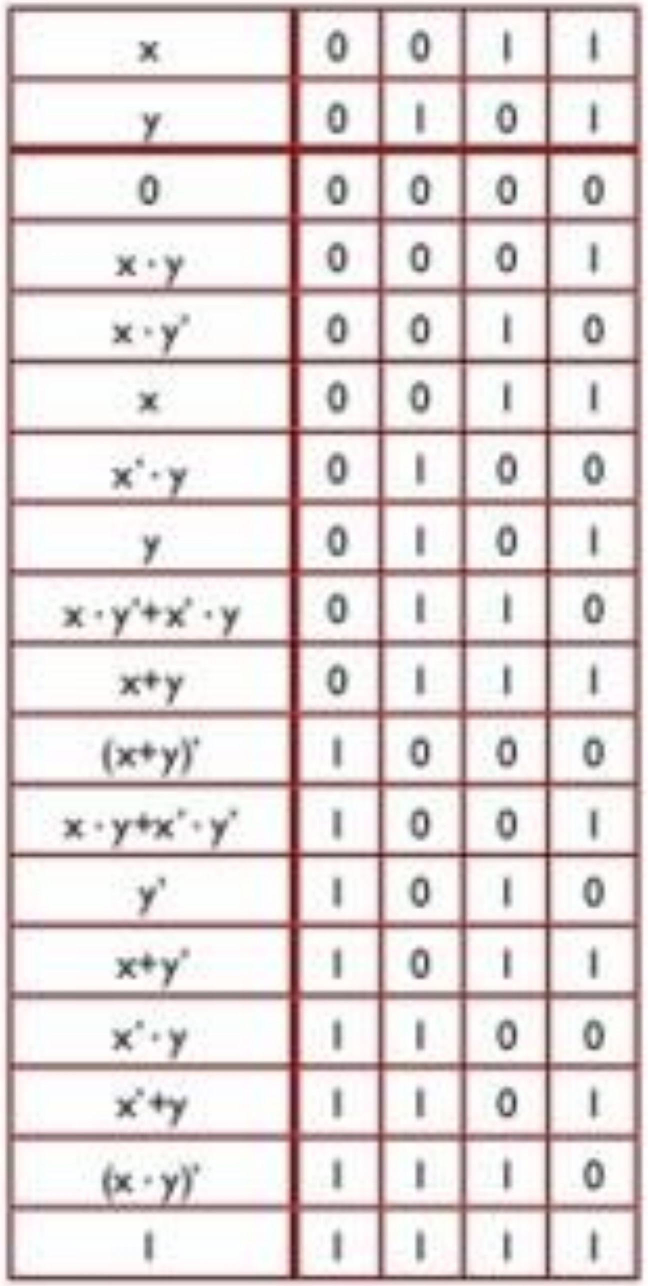
boolean m2 = (x % 2 == 0); // Set S2 = [0, 2, 4, 6, 8, 10, ...]

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





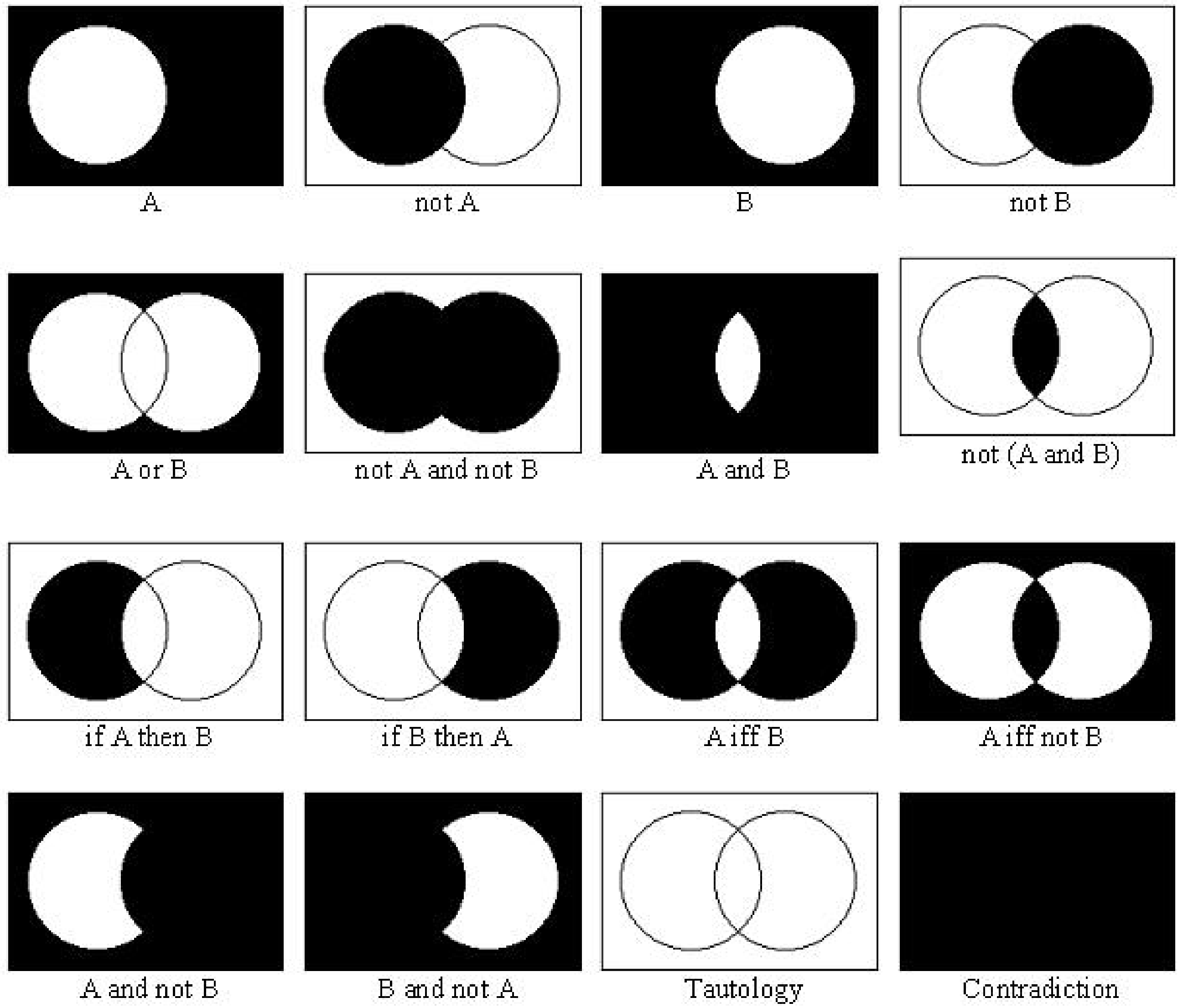


x	0	0	1	1
y	0	1	0	1
0	0	0	0	0
$x \cdot y$	0	0	0	1
$x \cdot y'$	0	0	1	0
x	0	0	1	1
$x' \cdot y$	0	1	0	0
y	0	1	0	1
$x \cdot y' + x' \cdot y$	0	1	1	0
$x+y$	0	1	1	1
$(x+y)'$	1	0	0	0
$x \cdot y + x' \cdot y'$	1	0	0	1
y'	1	0	1	0
$x+y'$	1	0	1	1
$x' \cdot y$	1	1	0	0
$x'+y$	1	1	0	1
$(x \cdot y)'$	1	1	1	0
1	1	1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

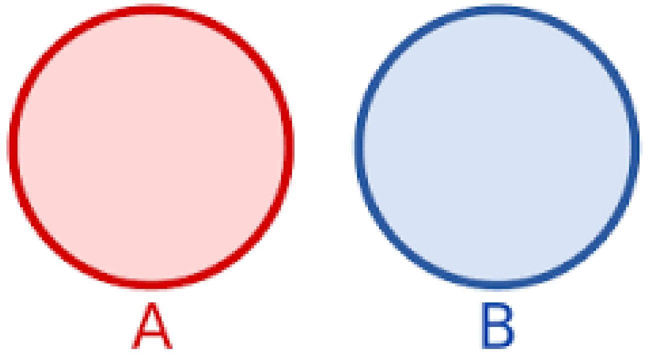
Advanced Venn Diagram for all 16 Logic Functions for Two Inputs *f(A, B)*



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 contained by (x>1)

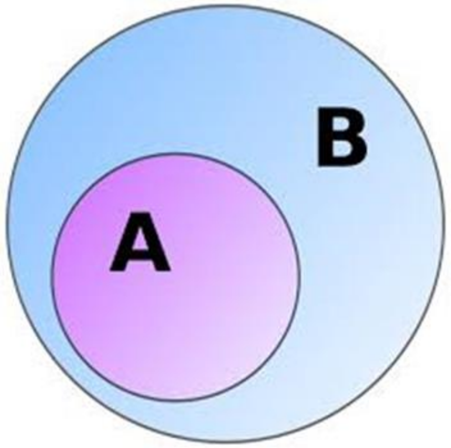
If you want to have A test first and B-A test:

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

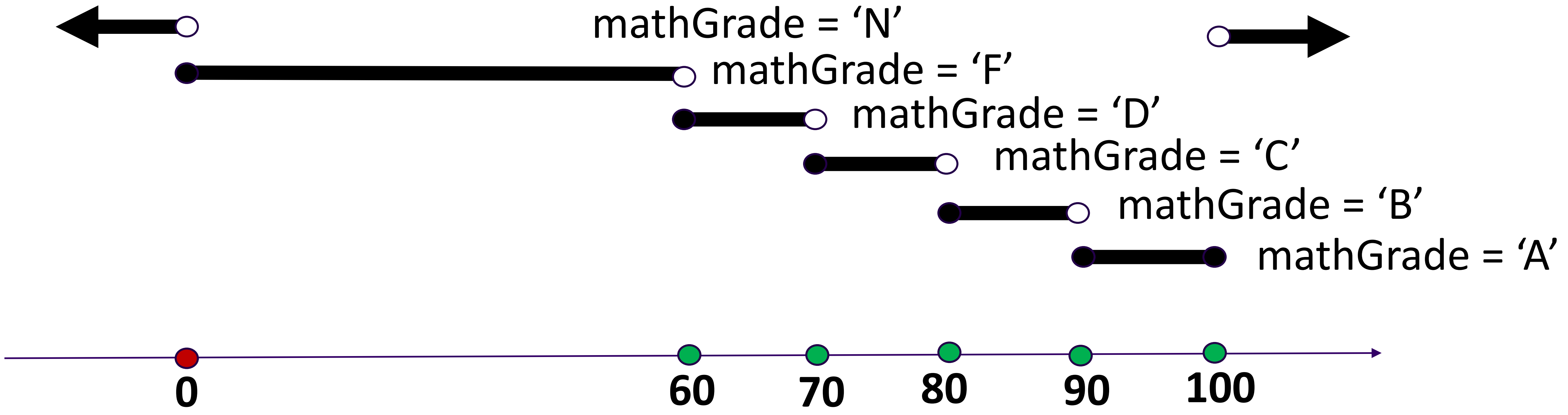
If you want to have A test and then B test:

```
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)



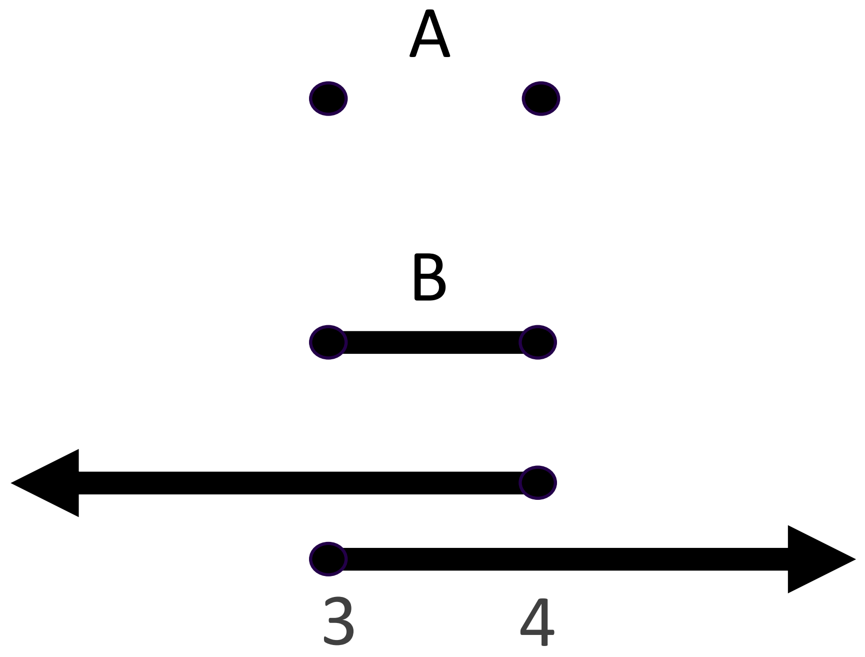
Sometimes the Logic Design Result Also Depends on the Data Type as Well

```
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



Lab

CALENDAR YEAR

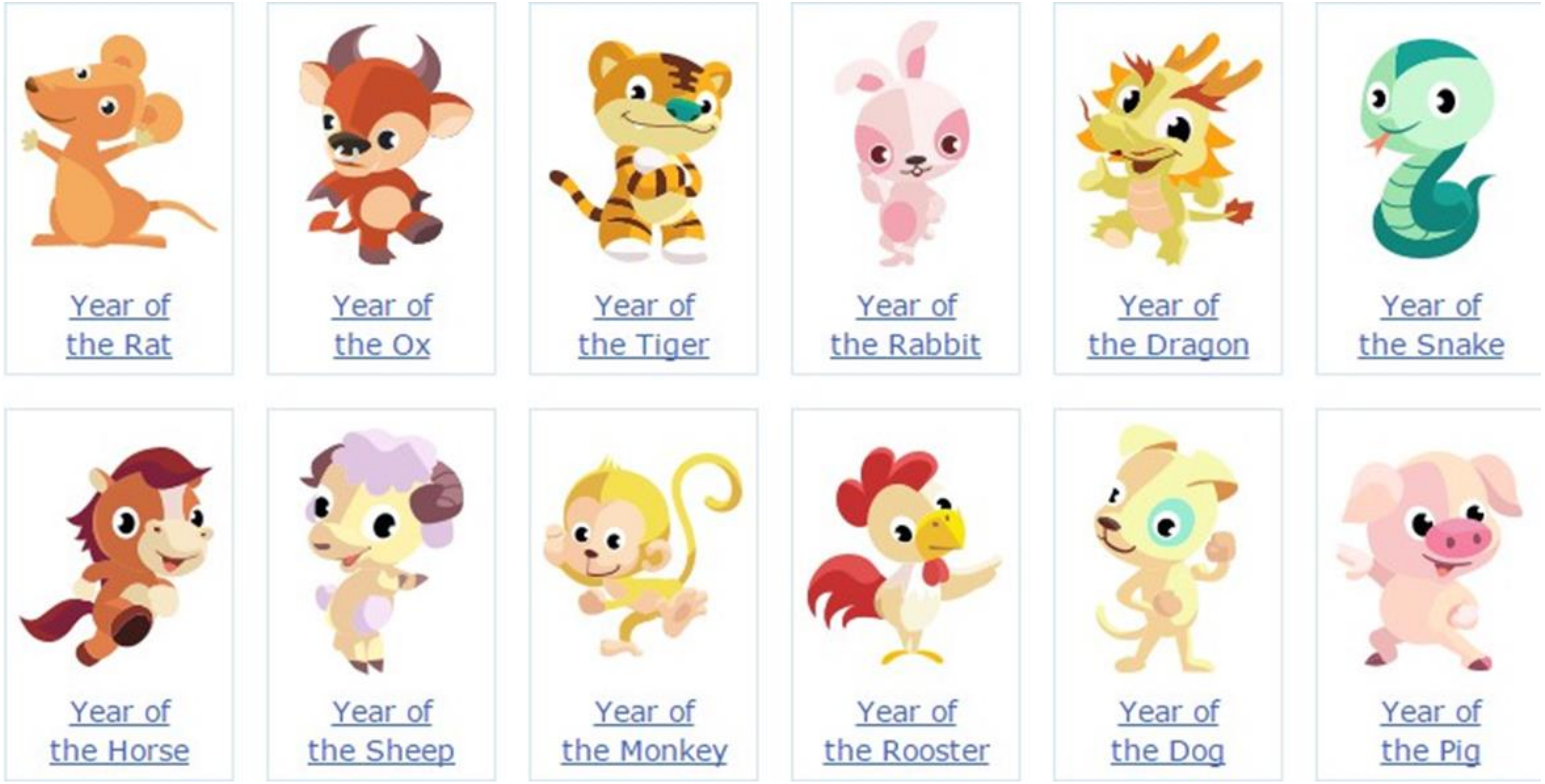
LEAPYEAR.JAVA

Background Information: (Leap Year)

- Which Years are Leap Years?
- In the Gregorian calendar 3 criteria must be taken 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

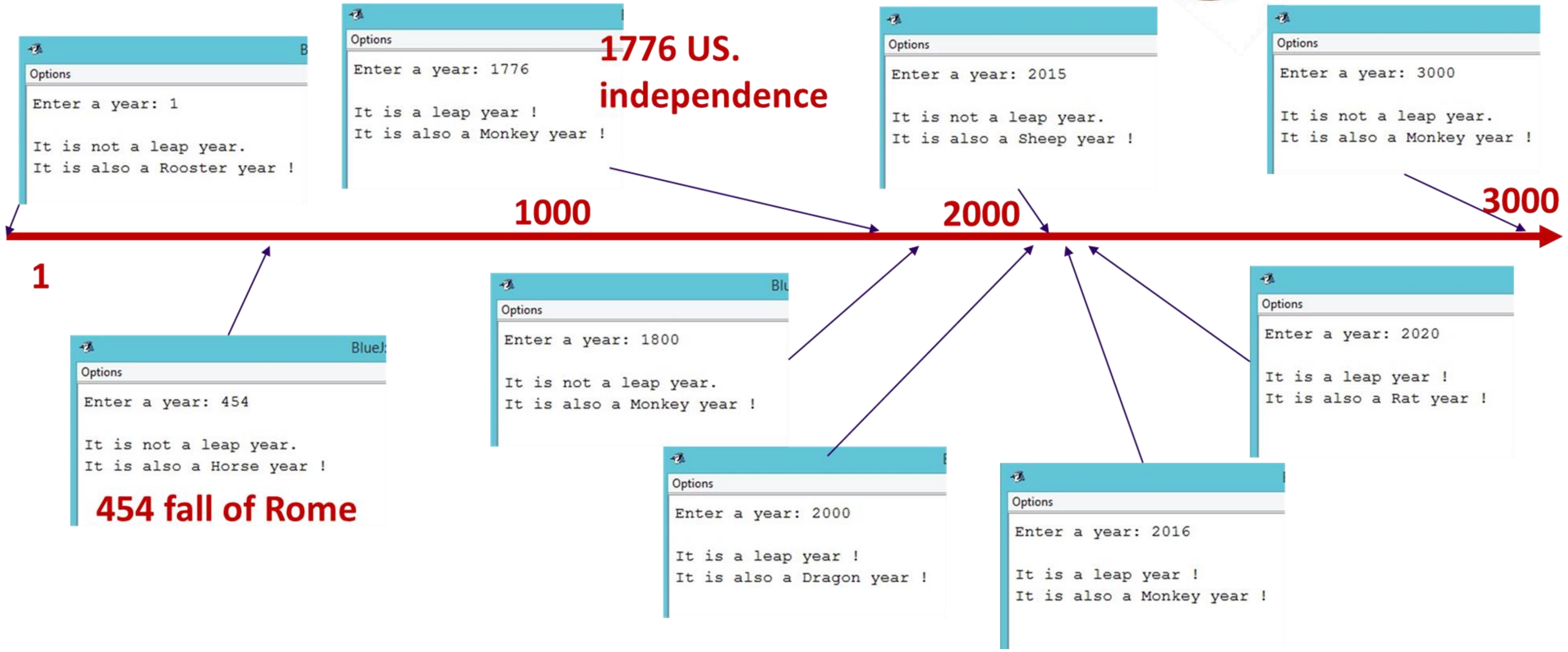
Background Information:
(Chinese Zodiac)

- 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:



```
Enter a year: 1

It is not a leap year.
It is also a Rooster year !
```

```
Enter a year: 454

It is not a leap year.
It is also a Horse year !
```

```
Enter a year: 1776

It is a leap year !
It is also a Monkey year !
```

```
Enter a year: 1800

It is not a leap year.
It is also a Monkey year !
```

```
Enter a year: 2000

It is a leap year !
It is also a Dragon year !
```

```
Enter a year: 2015

It is not a leap year.
It is also a Sheep year !
```

```
Enter a year: 2016

It is a leap year !
It is also a Monkey year !
```

```
Enter a year: 2020

It is a leap year !
It is also a Rat year !
```

```
Enter a year: 3000

It is not a leap year.
It is also a Monkey year !
```