IDENTIFIERS

CSC 111

Outline

1. Identifiers

2. Memory Space

3. Data Types

4.Data state

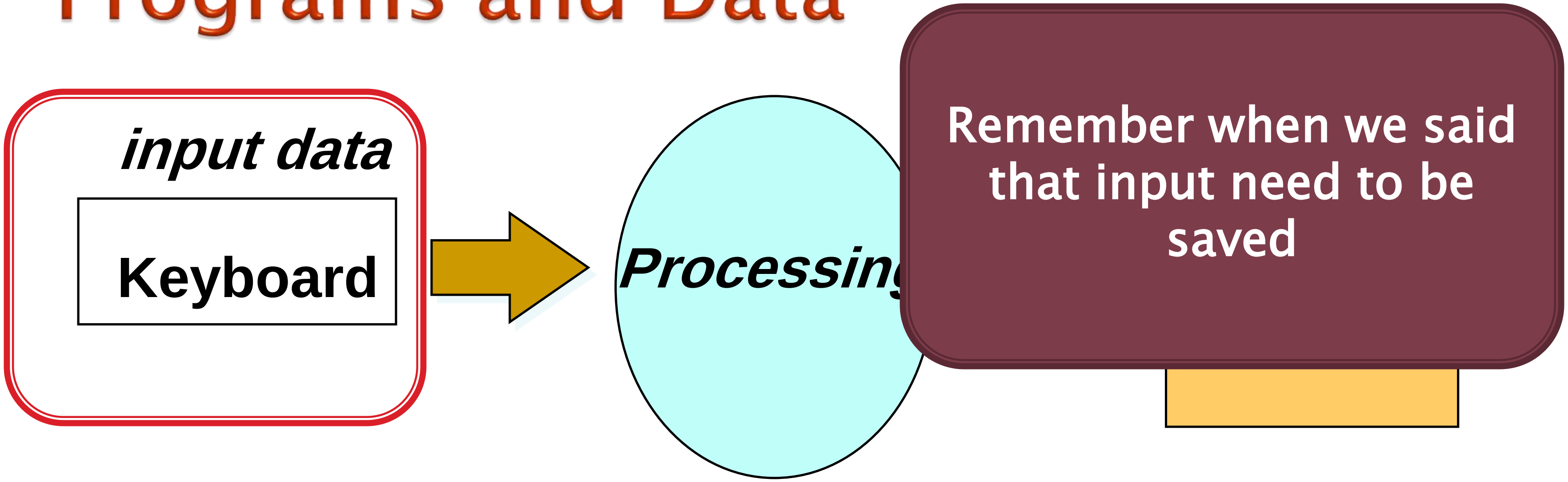
5. Declaration

    5.1 Constants

    5.2 Variables

6. Examples

Programs and Data



Remember when we said that input need to be saved

- Most programs require the temporary storage of data.
- The data to be processed is stored in a temporary storage in the computer's memory: memory space .
- A memory space has three characteristics
  - **Identifier :** name for that space
  - **Data Type :** Specifies how much space to store in memory
  - **State : is it variable ? or Constant**

1. IDENTIFIERS

- Identifiers are names of things such as:
  - Methods: a set of processing operations
  - Classes

- Rules for identifiers' names include:
  - May consist only of:
    - Letters (a – z or A – Z),
    - Digits (0 – 9),
    - Underscore (_),
    - Dollar sign ($)
  - Should NOT begin with a digit
  - Not a reserved word:
    - These are some words used in the Java language.
    - They are interpreted by the compiler to do a specific thing.
    - Examples of reserved words include: public, class, void, etc…

In our course, reserved words are written in light blue

Identifier names are case sensitive: number, Number, NUMBER represent three different identifiers.

1. IDENTIFIERS

EXAMPLES OF IDENTIFIERS’ NAMES

- The following identifiers’ names are valid:
  - First
  - payRate
  - $Amount
  - employee_salary
  - _Update

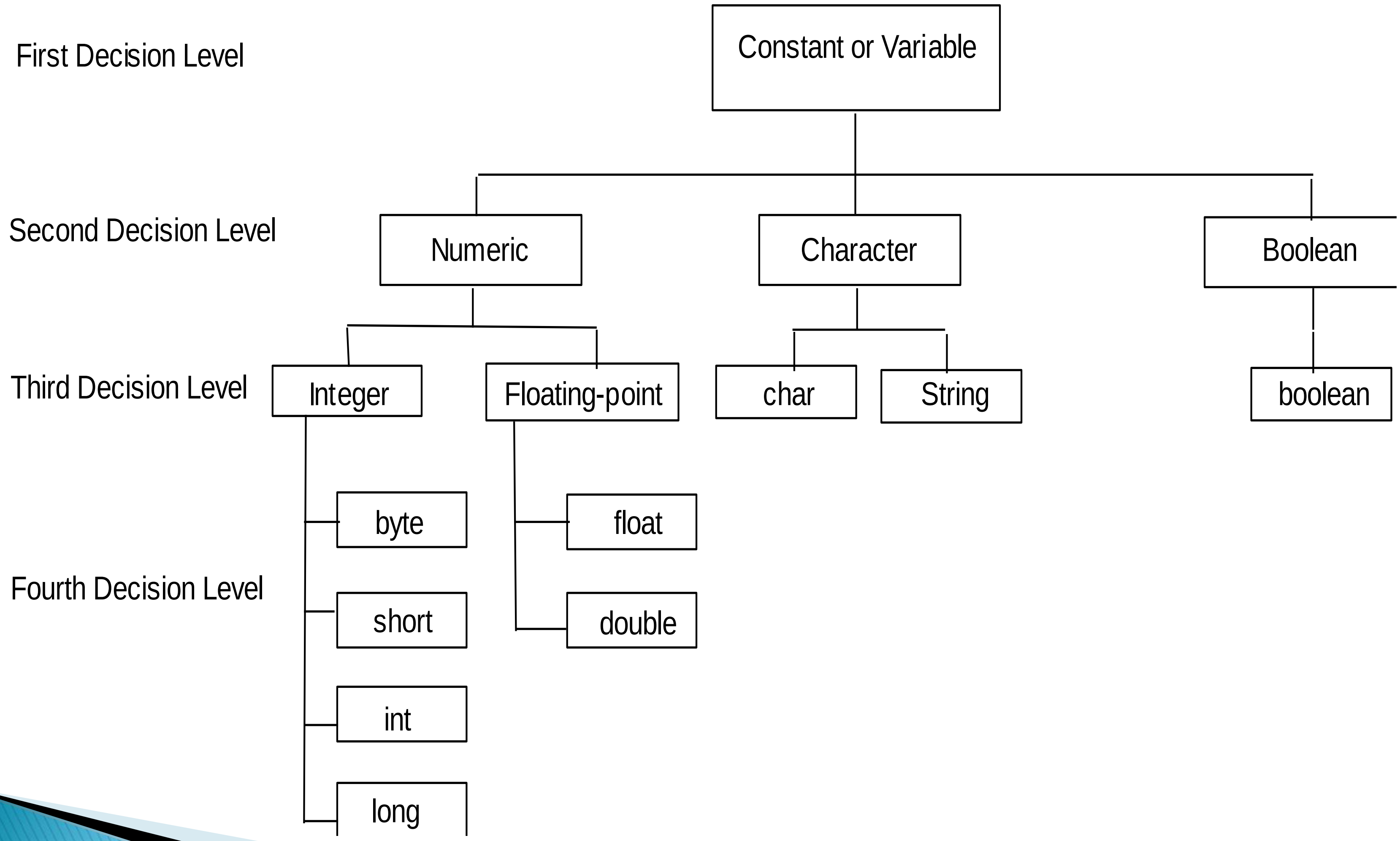
- The following identifiers’ names are NOT valid (illegal):

mid-term	- is an illegal character
add salary	space is an illegal character
one+two	+ is an illegal character
2nd	must NOT begin with a digit
public	Reserved word

2.Data Type

- The data type defines what kinds of values a memory space is allowed to store.
- All values stored in the same memory space should be of the same data type.
- All constants and variables used in a Java program must be defined prior to their use in the program.

Java built-in Data Types



2. DATA TYPES

PRIMITIVE DATA TYPES (1) – Integers

- Java categorizes integer data into the following primitive types:

Type	Size	Min. Value	Max. Value
byte	8 bits	- 128 = - 2^{7}	+ 127 = 2^{7} -1
short	16 bits	- 32,768 = - 2^{15}	+ 32,767 = 2^{15} - 1
int	32 bits	- 2,147,483,648 = - 2^{31}	+2,147,483,648 = 2^{31} - 1
long	64 bits	-9,223,372,036,854,775,808 = - 2^{63}	+ 9,223,372,036,854,775,808 = 2^{63} - 1

- All above types store numbers with no decimal point: integers.
- Positive integers do not require a + sign in front of them
- No commas are allowed within integers
- Note that larger size implies lower minimum and higher maximum
- Words in blue are reserved words

2. DATA TYPES

PRIMITIVE DATA TYPES (2) – Decimals

- Java categorizes decimal (or real) data into the following primitive types:

Type	Size	Min. Value	Max. Value	Number of Significant bits
float	32 bits	- 3.4e+38	+ 3.4e+38	Up to 7 after the decimal point (Single precision)
double	64 bits	- 1.7e+308	+ 1.7e+308	Up to 15 after the decimal point (Double precision)

- All above types store numbers with decimal point: floating-point.
- In Java, real numbers are represented using the floating-point notation:

Number	Scientific Notation	Floating-point Notation
4387	$4.387 * 10^{3}$	4.387e+3
438791	$4.38791 * 10^{5}$	4.38791e+5
0.0005	$5.0 * 10^{-4}$	5.0e-4
0.0000265	$2.65 * 10^{-5}$	2.65e-5

2. DATA TYPES

PRIMITIVE DATA TYPES (3) – Character

Type	Size	Min. Value	Max. Value	Description
char	16 bits	0	65,535 = $2^{16} - 1$	Stores the Unicode of single characters

- Any key on the keyboard is represented by a char data type
- Values of char type are enclosed between single quotes such as:

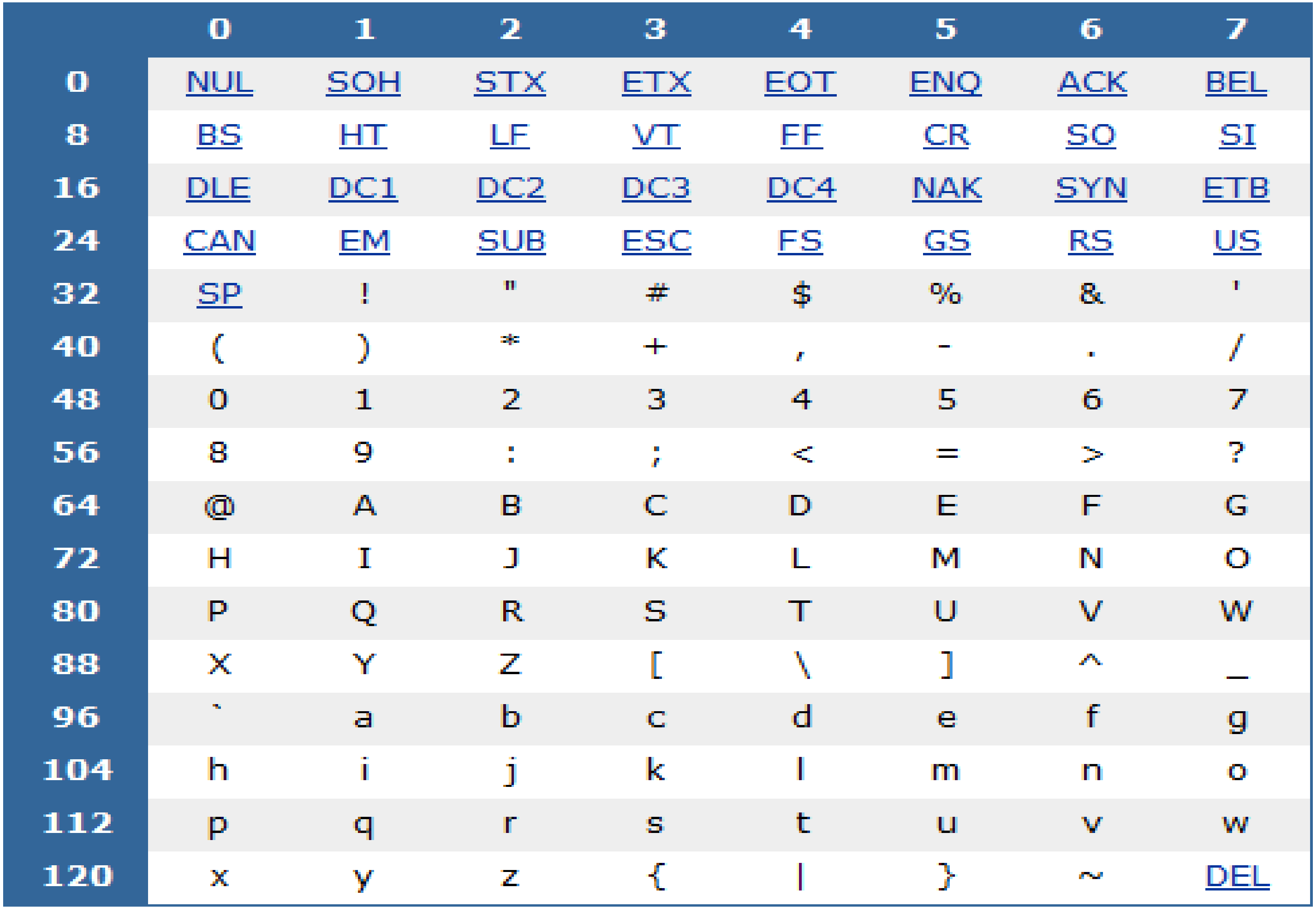
  'A' 'a' '%' '$' '*' '&' '7' ' '
- The following values can NOT be represented by a char type:

  'Abc' '>=' 'Fatma' ➔ these are rather string types (non-primitive).
- Note that the space may be represented as a char type
- Java uses the Unicode coding system to represent characters in memory. Each character has a unique code.
- There are 65,535 unique codes. For example, the value 65 corresponds to the letter 'A'; the value 97 represents 'a'; and so on. The table in the next slide represents all letters codes.

In our course, space character is represented by the letter '~'

2. DATA TYPES

The ASCII character set



	0	1	2	3	4	5	6	7
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
8	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
24	CAN	EM	SUB	ESC	FS	GS	RS	US
32	SP	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7
56	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G
72	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W
88	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g
104	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL

2. DATA TYPES

PRIMITIVE DATA TYPES (4) – Boolean

Type	Size	Min. Value	Max. Value	Description
boolean	1 bit	0	1	Stores either true or false.

- The boolean data type handles logical expressions that evaluate to either true or false.

Example of DATA TYPES

- Different programs deal with different data
  - An employee payroll program paycheck processes data such as:

Number of hours	(a fraction number)
Pay rate	(a fraction number)
Marital status	(a character)
Number of dependents	(a whole number)

  - A Registrar system in a university processes data such as:

GPA	(a fraction number)
Semester's load	(a whole number)

- Different data types support different operations
  - Numeric data are added, subtracted, multiplied, etc…
  - Character data are sorted, concatenated, etc…

3-State of the Memory Space

- The state of the memory space is the current value (data) stored in the memory space.

- The state of the memory space:
  - May be changed.
    - In this case the memory space is called variable.
  - Cannot be changed.
    - In this case the memory space is called constant.

Identifier Conventions in Java

- Constants:
  - All uppercase, separating words within a multiword identifier with the underscore symbol, _.
- Variables
  - All lowercase.
  - Capitalizing the first letter of each word in a multiword identifier, except for the first word.

4. DECLARATION

- Declaration allocates appropriate memory space to identifiers based on their types.
- Any identifier must be declared before being used in the program.
- The declaration of a variable means allocating a memory space which state <u>(value) may change.</u>
- The declaration of a constant means allocating a memory space which state <u>(value) cannot change.</u>

4.1 Constant Declaration

```
final dataType constIdentifier = literal | expression;
```

```
final double PI               = 3.14159;
final int    MONTH_IN_YEAR    = 12;
final short  FARADAY_CONSTANT = 23060;
```

The reserved word **final** is used to declare constants.

These are constants, also called *named constant*.

These are called *literals.*

```
final int MAX        = 1024;
final int MIN        = 128;
final int AVG        = (MAX + MIN) / 2;
```

This is called *expression.*

4.2 Variable Declaration

- A variable may be declared:
  - With initial value.
  - Without initial value.

- Variable declaration with initial value;

```
dataType variableIdentifier = literal | expression;
  double avg             = 0.0;
  int    i                = 1;
  int     x =5, y = 7, z = (x+y)*3;
```

- Variable declaration without initial value;

```
dataType variableIdentifier;
  double avg;
  int    i;
```

4.2 VARIABLES DECLARATION

- In Java, double is the default type of a floating-point number.
- When using float literals, the number should be written as shown below; otherwise, the compiler would give an error message (syntax error) :

Example 5

```
float x=5.33f;
float length=12.33f, width= 6.333f, radius=0.3f;
```

5. EXAMPLES – PROGRAM 1

```
// This example illustrates data declaration & manipulation
// program to calculate area of circule
// import necessary libraries
public class dataManipulation
{

   public static void main (String[] args)
      {
         // Declaration section: to declare needed variables
            double radius= 2.5, area;
         // Input section: to enter values of used variables
         // Processing section: processing statements
            area = PI * radius * radius;
         // Output section: display program output
            System.out.println (“The area of the circle of radius “ +
radius + “ is “ + area);
      } // end main
} // end class
```

Note: we only add static since declaration is outside main

➢ Program Output:

```
The area of the circle of radius 2.5 is 19.6349375
```

5.

```
// This example illustrates data declaration & manipulation
// import necessary libraries
public class dataManipulation
{
   public static void main (String[] args)
      {
         // Declaration section: to declare needed variables
            int num1= 10, num2 = num1 – 1;
            double sale = 0.02 * num1;
            char first;
         // Input section: to enter values of used variables
         // Processing section: processing statements
            first = ‘D’;
        // Output section: display program output
            System.out.println (“num1= “ + num1);//line output 1
            System.out.println (“num2= “ + num2);//line output 2
            System.out.println (“sale= “ + sale); //line output 3
            System.out.println (“first= “ + first); //line output 4
      } // end main
} // end class
```

```
num1= 10
num2= 9
sale= 0.2
first= D
```

Print statement either display a **text as it is inside double quotation “ “**
OR
Display the **value of a variable**

5. EXAMPLE

IMPORTANT NOTES

ANY VARIABLE MUST BE DECLARED BEFORE BEING USED

VARIABLES ON THE RIGHT HAND SIDE OF AN EQUATION SHOULD ALREADY HAVE VALUES

ALSO, VARIABLES THAT ARE TO BE PRINTED SHOULD ALREADY HAVE VALUES

VARIABLES GET VALUES EITHER BY:

1) INITIALIZATION,
2) CALCULATION,
3) INPUT FROM THE USER

Self-Check Exercises (1)

- Which of the following identifiers are illegal? Explain why:
  - God Father
  - &currency
  - final
  - 901
  - 4ever
- Write a program that converts from ^{0}C to ^{0}F.
- Write a program that adds two numbers.
- Write a program that calculates the average of three numbers.

Self-Check Exercises (2)

- Detect the errors in the following program:

```
public class FindError
{
   static final CENTIMETERS_PER_INCH = 2.54;
   public static void main (String[] args)
      {
            double inches;
            cm = CENTIMETERS_per_INCH * inches;
            System.out.println (“There are “ + cm + “cm in “ +
inches + “inches”);
      }
}
```