Programming Fundamentals

Learning Objectives

- Understand Number Systems
- Convert binary, hexadecimal numbers to decimal, and decimal to binary or hexadecimal numbers
- Understand representation of letters and numbers in a computer’s memory
- Discuss data type, and character strings
- Declare identifiers, constants, and variables
- Write and interpret assignments in a Java program
- Get acquainted with simple console data input and output

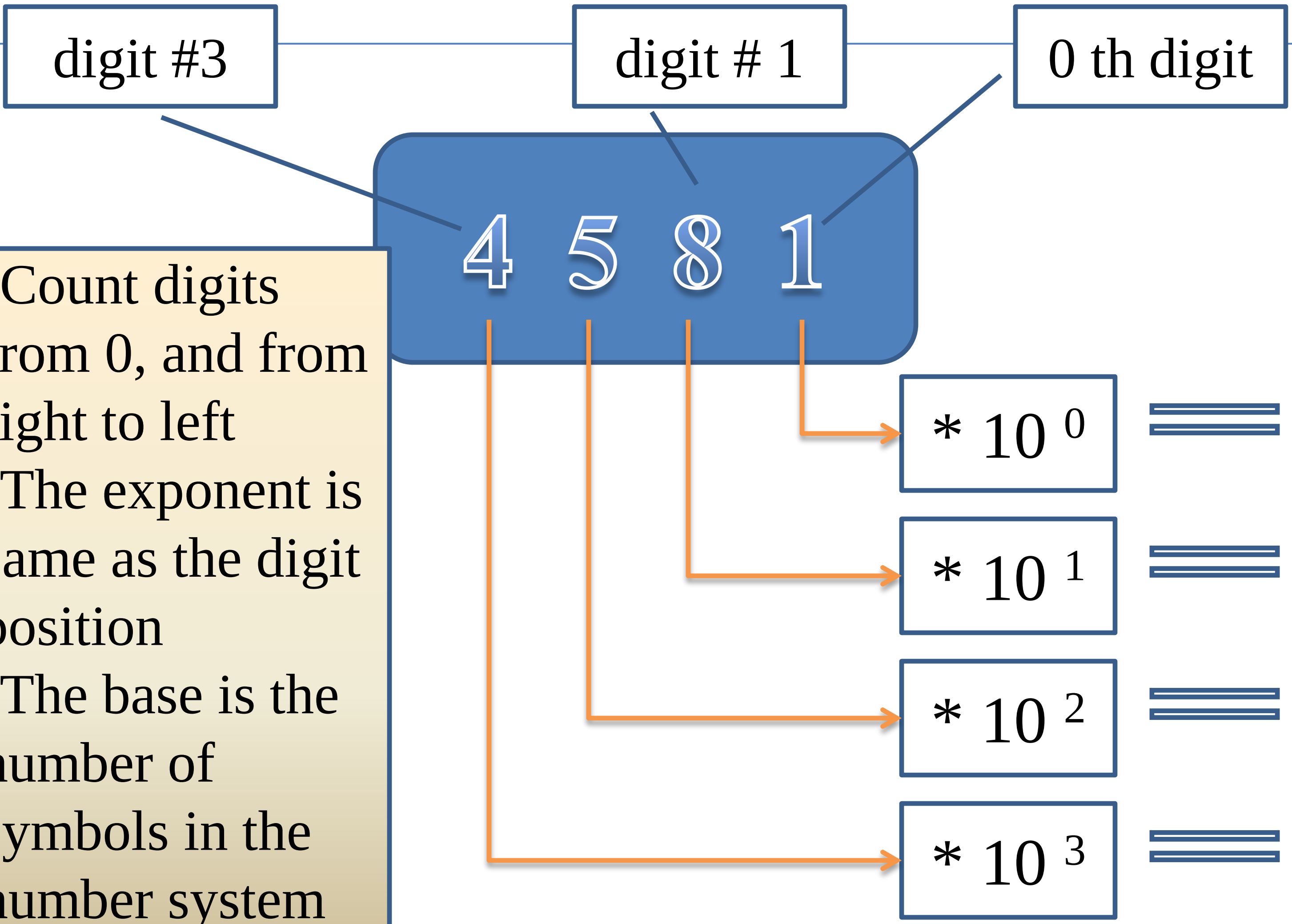
Bits, and Bytes

- Bit – Binary Digit
  - Binary or two
  - Two binary digits: 1 or 0
  - 1 is ON
  - 0 is OFF
- Byte – a group of bits organized into a group
  - An 8-bit group
  - Commonly used to represent a letter, number or a symbol
  - Examples: 1 , & W N – “

Decimal Numbers

- Trick question: how do you read the following number?
  - 4581
  - Can this number be represented using ***Place Value***?
  - Yes: $4*10^3+ 5*10^2+8*10^1+1*10^0$
- The Decimal numbers have 10 symbols
- Base of Decimal numbers = 10

Decimal Numbers



- Count digits from 0, and from right to left
- The exponent is same as the digit position
- The base is the number of symbols in the number system

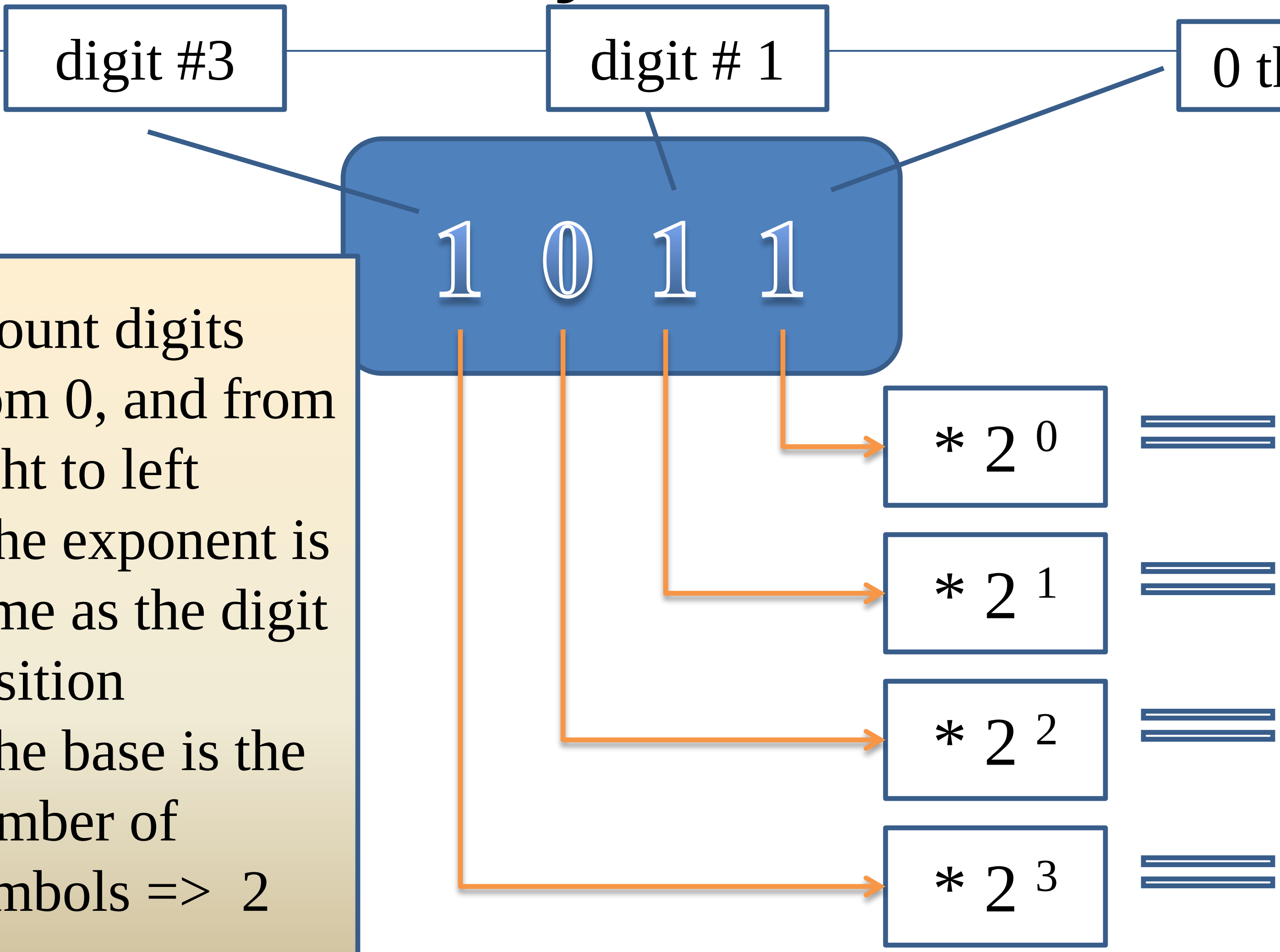
Binary Numbers

- Two digits; 1 0
- Base is 2
- Example of numbers

  01101001

  00101011
- What are these numbers in the example?
- Not used to Binary numbers
- Convert a Binary number to a Decimal number

Binary to Decimal



- Count digits from 0, and from right to left
- The exponent is same as the digit position
- The base is the number of symbols => 2

Binary to Decimal

01101001 => Value in decimal?

$1 * 2^0 = 1$

$0 * 2^1 = 0$

$0 * 2^2 = 0$

$1 * 2^3 = 8$

$0 * 2^4 = 0$

$1 * 2^5 = 32$

$1 * 2^6 = 64$

$0 * 2^7 = 0$

Total = 105

Digits are Right to left

YOUR TURN

00101011

10110011

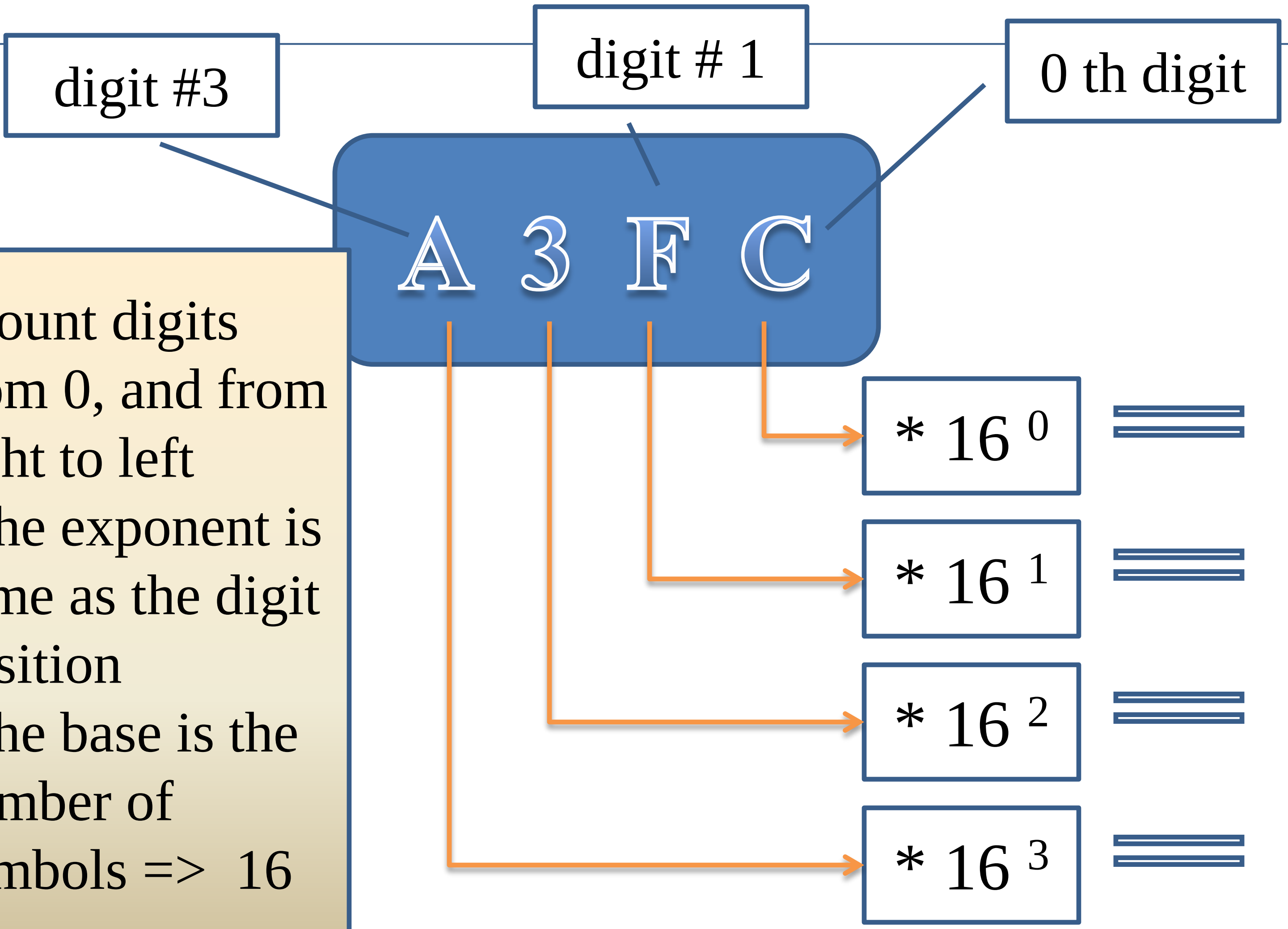
Other Number Systems

- Hexadecimal – 16 symbols
  - Base is 16
  - Symbols: 0 1 2 3 4 5 6 7 8 9 A B C D E F
  - Decimal equivalent: 0 through 9, no change

A is 10	B is 11	C is 12
D is 13	E is 14	F is 15

  - Hex (short form of Hexadecimal) can be converted to Decimal
    - similar to Binary to Decimal conversion

Hexadecimal to Decimal



- Count digits from 0, and from right to left
- The exponent is same as the digit position
- The base is the number of symbols => 16

Other Number Systems

- Octal – 8 symbols
  - Base is 8
  - Symbols: 0 1 2 3 4 5 6 7
  - Decimal equivalent: 0 through 7, no change
- Conversion to Decimal is similar to others
- Avoid confusion by indicating number system

176 (Decimal)

176 (Octal)

176 (Hex)

These three numbers look same but have very different values

Decimal to Binary

1. Divide the decimal number by 2 and note the quotient and the remainder
2. The remainder is the right most digit
3. Divide the quotient by 2 and note the quotient and the remainder
4. The remainder is the next right most digit
5. If the quotient is 1 or 0, it is the last digit and you are done
6. If the quotient in step 5 is not 1 or 0, go back to step 3 and repeat the process

Decimal to Binary: Example

1. Convert 172 decimal to binary

172 ÷ 2 => Quo 86 Rem 0	0
86 ÷ 2 => Quo 43 Rem 0	00
43 ÷ 2 => Quo 21 Rem 1	100
21 ÷ 2 => Quo 10 Rem 1	1100
10 ÷ 2 => Quo 5 Rem 0	01100
5 ÷ 2 => Quo 2 Rem 1	101100
2 ÷ 2 => Quo 1 Rem 0	**10**101100

Decimal to Binary

- Your turn to convert these decimal numbers to equivalent binary:
  - 211
  - 149
  - 307

Character Sets

- How many unique patterns are created using
  - 4 bits ?
  - 8 bits ?
  - 16 bits ?
- Unicode - a 16 bit pattern
- The first 128 characters are ASCII
- Java uses Unicode

Character Set: Examples

Character	Decimal Value	Binary Value (7 bits)
a	97	1100001
A	65	1000001
6	54	0110110
$	36	0100100

- Suppose a machine can make sense of the binary values
- Is the value of ‘a’ same as ‘A’ ?
- Is the value of ‘6’ more or less than ‘$’ ?
- Which is bigger, ‘a’ or ‘A’ ?

Representing Numbers

- Numbers are represented in binary

  Example:

  6 in 7 bit binary 0000110

- Digit 6 as a character (from previous slide)

  6 => Decimal 54 => 0110110

- Is number 6 same as the character 6?
- Character 6 is shown in a program as ‘6’
  - Characters are surrounded by a pair of single quotes

Representing Strings

- Strings are a set of characters
- The representation shown here is conceptual

String	Representation (In Hex)
Apple	4170706C65
GOOD	474F4F44
Hi4$	48693424

- Left to right: each two digits represent a character
- Example: A is 41, p is 70, l is 6C
- In a program, strings are written within a pair of quotes: “Brenda”

Identifiers

- Names of elements that appear in a program

```
// This is my first program
public class HelloWorld {
    public static void main(String [] args) {
       System.out.println("Hello World");
       System.out.print("Hello Universe");
       System.out.print("\nHello Rover");
    }
}
```

- All items in Red are identifiers
- Identifiers are created by the programmer

Identifiers

- Names of elements that appear in a program

```
import java.util.Scanner;
public class ComputeAverage {
    public static void main(String [] args) {
       Scanner input = new Scanner(System.in);
       double number1 = input.nextDouble();
       double number2 = input.nextDouble();
       double average = (number1 + number2) / 2.0;
       System.out.println("Average is "+average);
    }
}
```

- All items in Red are identifiers
- Identifiers are created by the programmer

Identifiers: Rules

- Can be a combination of alphabets, digits, underscore (_), and dollar sign ($)
- Can be of any length
- The first character may not be a digit
- No space can be used in an identifier
- It cannot be true, false, or null
- Rules for this class: check style guide

Keywords

- These are predefined by Java as reserved words
- Keywords can ONLY be used for the purpose designated by the language
- Examples:

– double	null	if
– class	private	public
– void	for	while
– char	true	int

Variables

- What do these PO Boxes got to do with variables?
- These are analogous to memory locations

Picture by jerebu, http://www.flickr.com/photos/jburgin/3652827846/sizes/m/in/photostream/

Variables

- Represents a memory area in the computer memory
- Variable name is an identifier
- A variable is used to store a specific **type** of data: integer, character, real number, string
- Needs to be declared before use

  type identifier;

Variables

Declaration	Data type	Example values
int homeworkScore1;	Only Integers	20 -34 456 0
float extraPerson;	Small real numbers	234.54 -3678.09
double priceOfTickets;	Large real number	234.54 -3678.09
String studentName;	Strings of characters	“Programming”
char firstInitial;	A character	‘A’ ‘K’ ‘C’
boolean isFinished;	A true/false data type	false true

- Memory is allocated when the program is successfully compiled
- There are potential bad consequences if data and types do not match

Variables

- Variables can be declared and given an initial value

  type identifier = expression;

- Initial values are assigned to the variable during the compile time
- We can also assign values to variables during the run time (discussed later)
- Examples:

```
int homeworkScore1 = 0;
char firstInitial = ‘A’;
```

Predefined Data Types: Integers

- Java provides for many integer data types
- When and why we use different integer types?

Data type	Range of values	Used for
byte	-128 to 127	Small integers, can be negative
int	Large (-2^{31} to $2^{31}-1$)	Signed integers
long	Very large (-2^{63} to $2^{63}-1$)	Very large signed integers
float	32 bit signed value	Large signed real numbers
short	-32,768 to 32,767	Signed small integers
double	64 bit signed values	Very large signed real numbers

Predefined Data Types: Integers

- Examples:

```
byte roomNumber = -128;
char middleInitial = ‘K’;
int studentCount;
int numberOfExams = 4;
long distanceToMars;   // Legal, but bad choice?
```

Predefined Data Types: real

- Examples:

```
float totalAmount = 23.57f; //f  (or F) is needed
        // the suffix f makes 23.57 a float
        // the default type of 23.57 is double
double averageScore = 70.0;
double gradePointAverage;
double roomRent;
```

Variable declaration in a Program

```
public class RoomSizeCalculator
{
    public static void main(String[] args)
    {
        double length = 37;     //represents length of a room
        double width = 22;      //represents width of a room
        double area;            //represents area of the room

        area = length * width;
        System.out.println("Room area is " + area);
    }
}
```

Declaring Strings

- Strings are a series of text characters
- Example:

```
String studentName;
String courseName = “Programming 1”;
        // double quotes are required
String twoLines = “Line1\nLine2”;
```

Variable declaration in a Program

```
public class RoomSizeCalculator
{
   public static void main(String[] args)
   {
      double length = 37;     //represents length of a room
      double width = 22;      //represents width of a room
      double area;            //represents area of the room
      String helloUser = "Hello Java Programmer";
      area = length * width;
      System.out.print(helloUser + '\n');
      System.out.println("Room area is " + area);
   }
}
```

Declaring Boolean Variables

- Boolean is based on true/false
- Java uses the key word ***boolean*** to indicate Boolean
- Example:

```
boolean professionalStudent;
boolean moreData = true;
boolean endOfAList = false;
```

Making Data Constant

- Declaration

```
final double TAX_RATE = 0.675;
final int SPEED = 70;
final char highestGrade = ‘A’;
final String className = “Programming 1”;
```

- Values of constants cannot be changed in the program
- General declaration format

final type identifier = expression;

Constant in a Program

```
public class RoomSizeCalculator
{
   public static void main(String[] args)
   {
      double length = 37;     //represents length of a room
      double width = 22;      //represents width of a room
      double area;            //represents area of the room
      final String HELLO_USER = "Hello Java Programmer";
      area = length * width;
      System.out.print(HELLO_USER + '\n');
      System.out.println("Room area is " + area);
   }
}
```

Assignment Statement

- Takes the form

```
variable = expression;
```

- Compare the above to type declaration
- Examples:

```
numberOfMinutes = 45;
count = 0;
firstInitial = ‘B’;
```

- Compare assignment of values between compile time and run time

Assignment in a Program

```
public class RoomSizeCalculator
{
   public static void main(String[] args)
   {
      double length;    //represents length of a room
      double width;     //represents width of a room
      double area;          //represents area of the room
      final String HELLO_USER = "Hello Java Programmer";

      length = 27;
      width = 18;
      area = length * width;
      System.out.print(HELLO_USER+ '\n');
      System.out.println("Room area is " + area);
   }
}
```

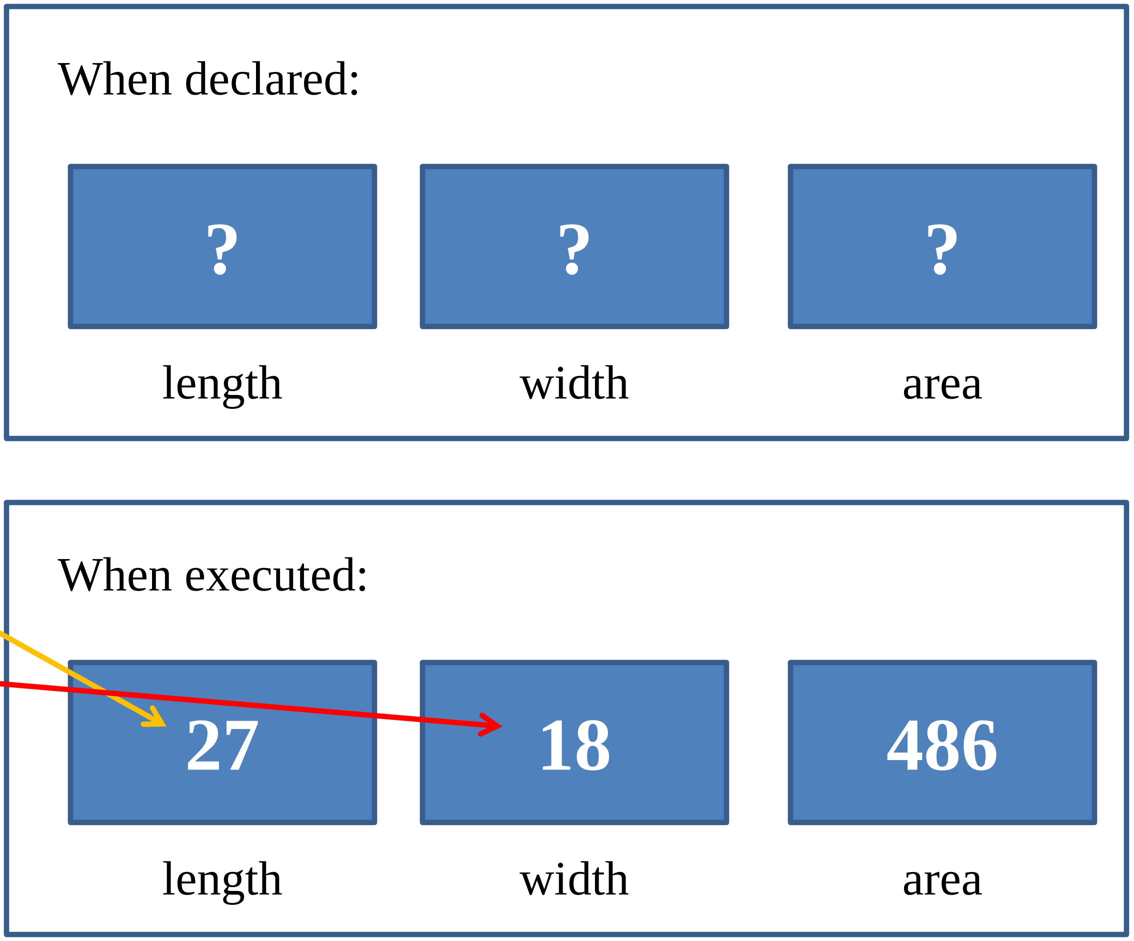
Basic Arithmetic Operation

Operator	Operation
*	Multiplication
/	Division
%	Remainder
+	Addition
-	Subtraction

Operator Precedence	
* / %	Left to right
+ -	Left to right

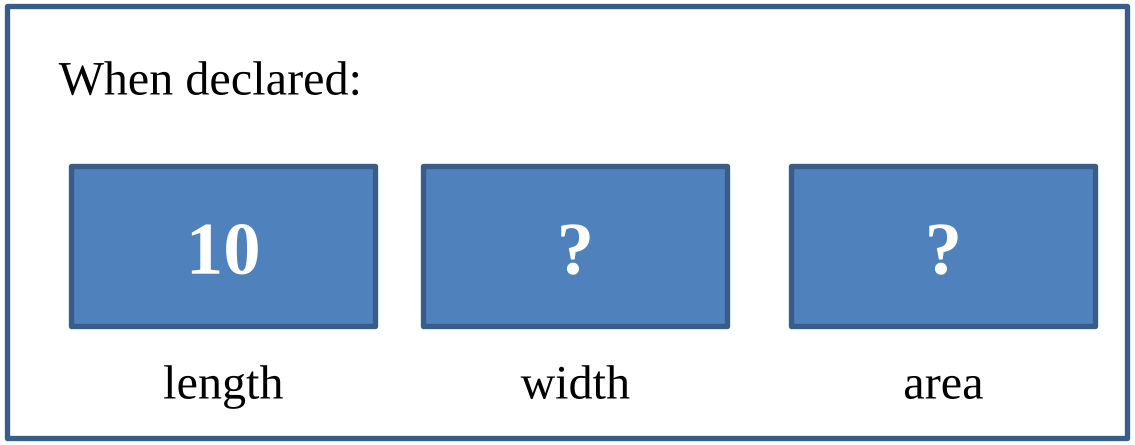
Expressions and Variables

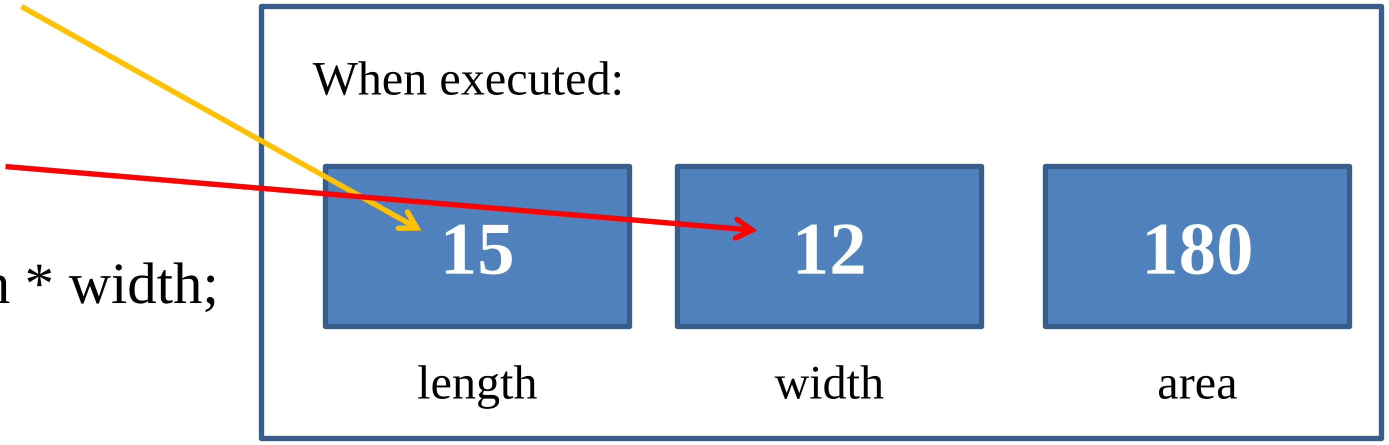
```
double length;

double width;

double area;

length = 27;

width = 18;

area = length * width;
```



Expressions and Variables

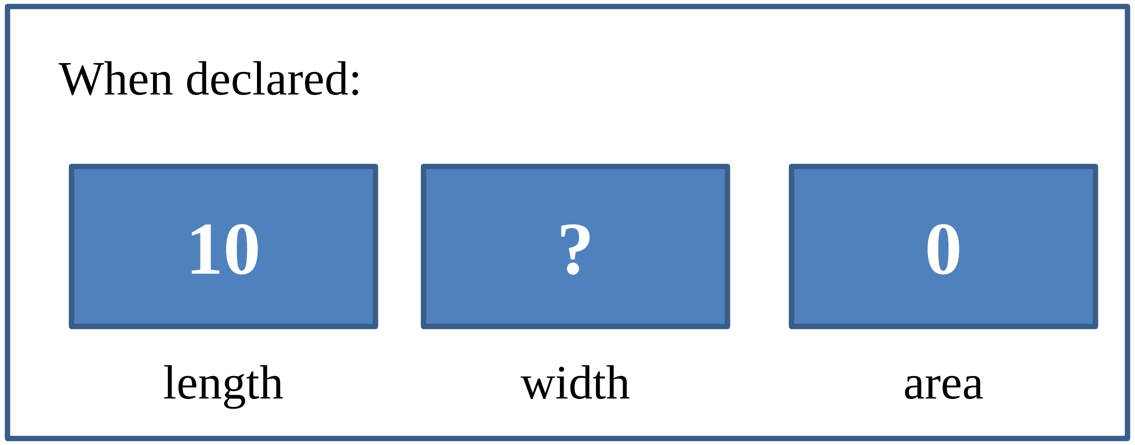
```
double length = 10;

double width;

double area;

 length = 15;

 width = 12;

 area = length * width;
```

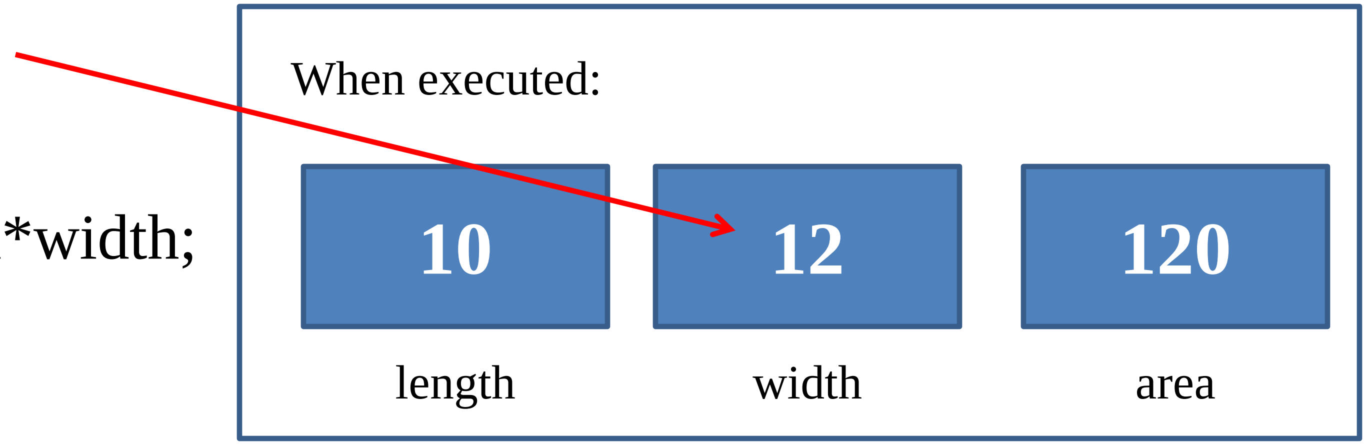




Expressions and Variables

```
double length = 10;

double width;

double area = 0;

 width = 12;

 area= length*width;
```





Expressions and Variables

```
int  num = 0;  // legal declaration
final int SPEED = 20;  // legal constant
num = 40;  //legal assignment;
40 = SPEED;      //Illegal, why?
SPEED = num;  //Illegal, why?
num = SPEED;  //Legal or Illegal ?
```

Mixed Expressions

- Occurs when operands are of different types

  2 * 7.9 – 5 + 67.99

- In general, smaller typed operands are promoted to larger type
- Next several slides show a few examples

Mixed Expressions

- Operands of the same type – No problem
- Operands are different types – Potential problems

(1) 89 + 76 / 3.25 ← Mixed int and double

All values converted to double

(2) 2 * 7.9 – 5 + 67.99

Mixed Expressions

- Assigning a double to an int is a problem
- The following generates syntax error

```
double answer = 10.0;
int smallValue;
smallValue = answer;  // syntax error
```

- The numeric range of an **int** is smaller than that of a **double**

Mixed Expressions

- An int can be assigned to double
- Conversion occurs during the assignment

```
double answer;
int smallValue = 10;
answer = smallValue;
```

  - ON assignment ***answer*** gets 10.0

Mixed Expressions

- Assignment does not always promote values

```
double answer;
answer = 10/3;
```

  - The conversion occurs only during the assignment
  - 10 / 3 is an integer division – Result is 3
  - ON assignment ***answer*** gets 3.0 not 3.33333

Generalizing Programs

Reading from Console

An Example Program

- Design an application that displays the number of square feet in a house. Declare and initialize the length and width of the house to 37 and 22 respectively.
- Question:
  - Where do we start?
  - The sequence of actions for the program
  - Variable to be declared, their types
  - Java code

Example Code

```
public class RoomSizeCalculator
{
    public static void main(String[] args)
    {
        double length = 37;     //represents length of a room
        double width = 22;      //represents width of a room
        double area;            //represents area of the room

        area = length * width;
        System.out.println("Room area is " + area);
    }
}
```

Program Limitations

- Program is good for a room with length =37 and width = 22
- What if your room is 15 x 12?
- Can we write a program that will work for any room size?
- Scanner class can be used to read next input value from the standard input

Scanner class

- Three standard streams are created by default for all programs
  - System.in – input bytes from standard input (often key board)
  - System.out – output to standard output (often console)
  - System.err – Output error message to screen
- Each of these can be redirected

Scanner class

- Java library defines a Scanner class that works with System.in to read text
- The Scanner class is stored in
  - java.util.Scanner
  - This library must be imported to the program
- We need to create an object of Scanner type before we can use it
  - Scanner input = new Scanner (System.in);

Scanner with System.in (example)

```
import java.util.Scanner;
public class ScannerTest {
    public static void main (String [] args) {
          Scanner input = new Scanner(System.in);
         // reads from standard input: int, float, double, boolean
          // Uses standard delimiter of spaces
          int number = input.nextInt();
          System.out.println("Input integer = "+number);
          float realValue = input.nextFloat();
          System.out.println("Input float = "+realValue);
          double doubleValue = input.nextDouble();
          System.out.println("Input double = "+doubleValue);
          ……
     }
}
```

Scanner methods

Method	Description
nextByte()	Reads an integer of the byte type
nextShort()	Reads an integer of the short type
nextInt()	Reads an integer of the int type
nextLong()	Reads an integer of the long type
nextFloat()	Reads a number of the float type
nextDouble()	Reads a number of the double type
next()	Reads a string that ends before a space
nextLine()	Reads a line of text until a new line