

# Lesson 5-2: Nested If Statements

Computer Science 46A: Introduction to Programming  
San José State University

# Announcements

- Homework #5 has been posted
- Lab #5 is tomorrow (3/1)

# Learning Objectives

By the end of this lesson, you should be able to:

1. Use nested if statements to construct decision trees
2. Use Scanner methods to check user input types

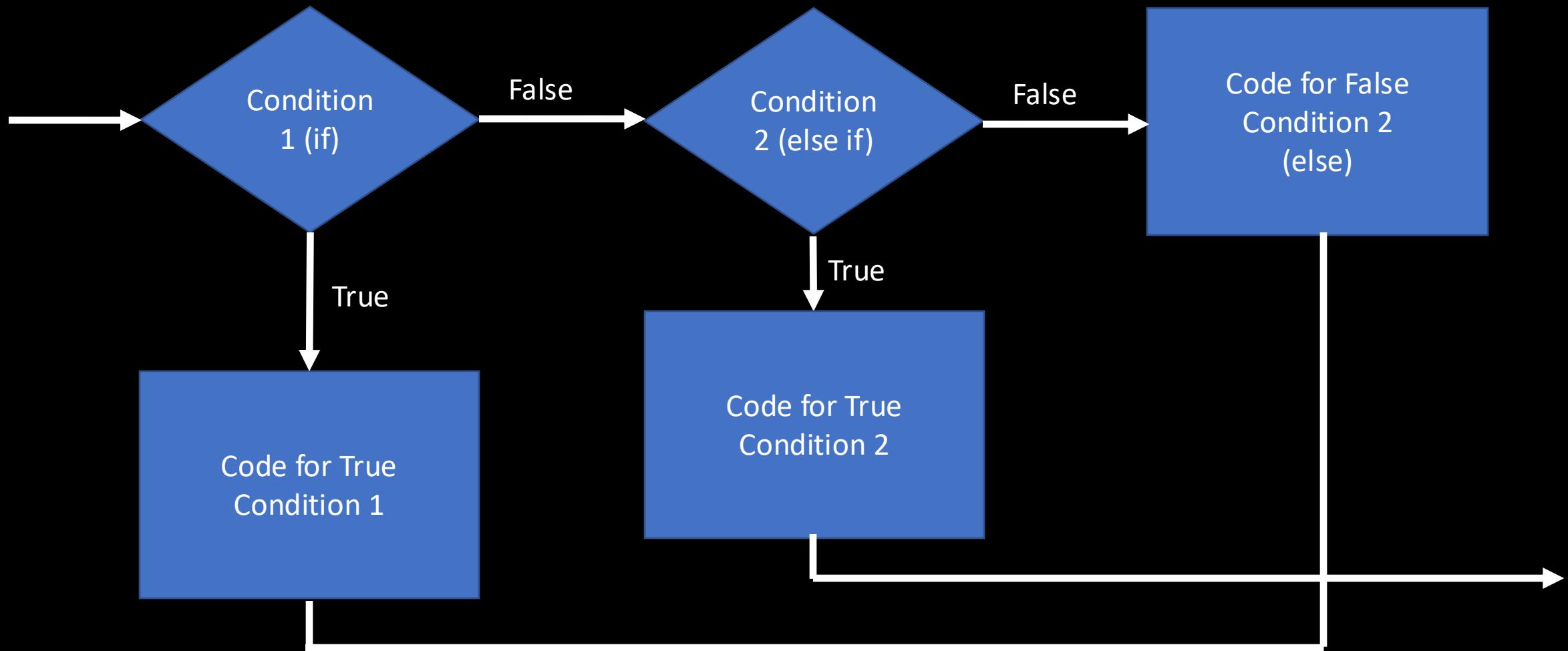
# Poll Everywhere: Question 1

```
if (4 > 5)
    System.out.print('A');
else if (5 == 5)
    System.out.print('B');
else if (4 < 5)
    System.out.print('C');
```

What will be the output?

- A) A
- B) B
- C) C
- D) B and C
- E) An Error

# If-else if-else Statement Diagram



# Nested if Statements

- A single if statement allows us to make a decision about our input
- What if we needed to make decisions based on those decisions?

The Answer: nested if statements

- A nested if statement is an if statement inside another if statement

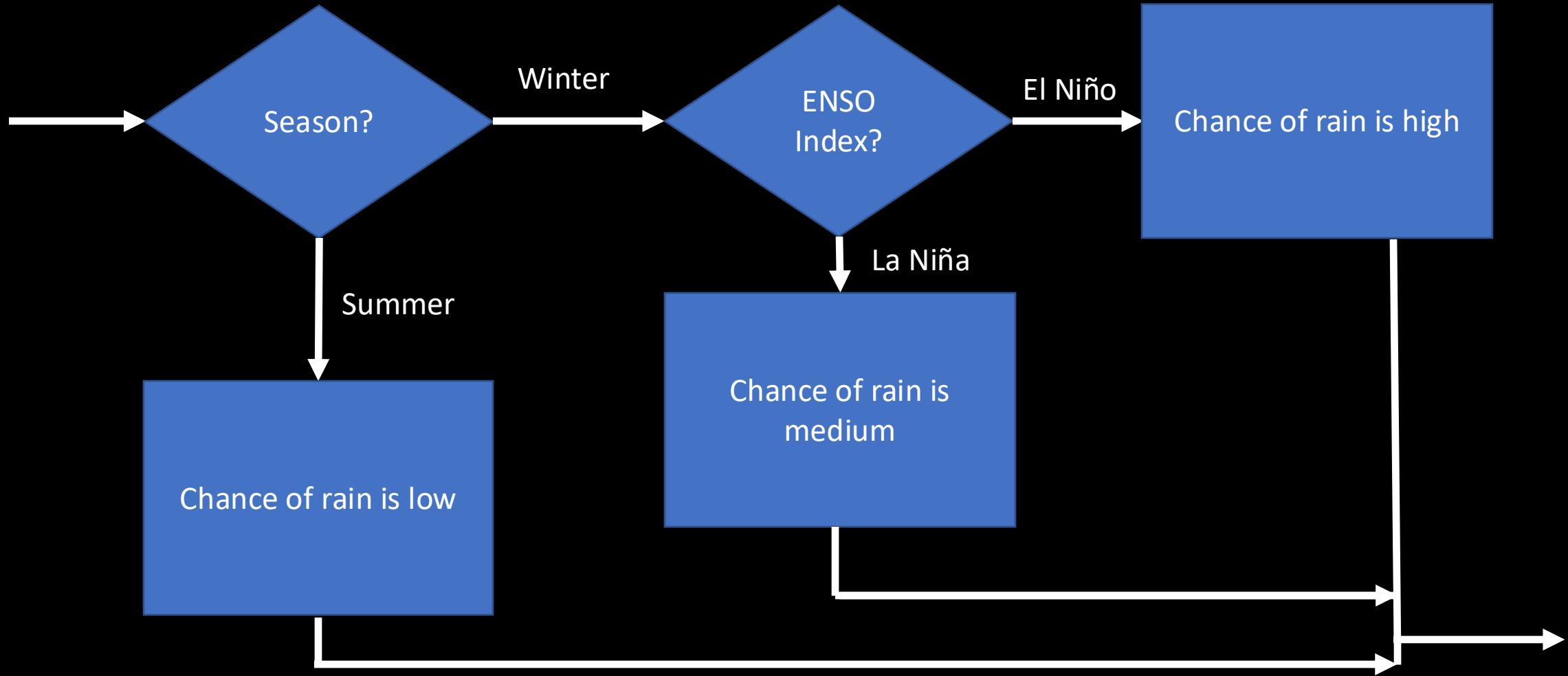
- Syntax

```
if (condition)
{
    if (another condition)
    {
        // code here
    }
}
```

# An Example from Climate: ENSO

- The “El Niño Southern Oscillation” (ENSO) is a climate pattern in the Pacific that has two phases: El Niño and La Niña
- Climate scientists quantify an ENSO index, which is positive for El Niño and negative for La Niña
- For California, these typically bring different types of weather
  - In Winter:
    - El Niño phase (positive index): more rainfall
    - La Niña phase (negative index): less rainfall
  - In Summer:
    - No discernible effect (very little rain in the summer)

# What's the chance of rain?



# Example: NestedENSO.java

Sample Output #1

```
Enter a season (winter or summer): winter
Enter the ENSO index: 1.5
During El Ni o in winter, the chance of rain is high
```

Sample Output #2

```
Enter a season (winter or summer): winter
Enter the ENSO index: -2.0
During La Ni a in winter, the chance of rain is medium
```

Sample Output #3

```
Enter a season (winter or summer): summer
Enter the ENSO index: 1.0
During summer, the chance of rain is low regardless of the ENSO phase
```

# Boolean Operators

- In the last example, we have used a nested if statement to check 2 different conditions
- What if we wanted to check the two conditions on the same line?
- The Answer:
  - Boolean operators:
    - The *and* condition: `&&` - is true if **both** conditions are true
    - The *or* condition: `||` - is true if **either** condition is true

# Poll Everywhere: Question 2

```
int x = 5;  
int y = 10;
```

Which of the following is true?

- A)  $(x > 7 \ \&\& \ y > 7)$
- C)  $(x < 7 \ \&\& \ y < 7)$
- B)  $(x < 7 \ \mid\mid \ y < 7)$
- D)  $(x > 7 \ \mid\mid \ y < 7)$

# Example: BooleanENSO.java

Sample Output #1

```
Enter a season (winter or summer): winter
Enter the ENSO index: 1.5
During El Ni o in winter, the chance of rain is high
```

Sample Output #2

```
Enter a season (winter or summer): winter
Enter the ENSO index: -2.0
During La Ni a in winter, the chance of rain is medium
```

Sample Output #3

```
Enter a season (winter or summer): summer
Enter the ENSO index: 1.0
During summer, the chance of rain is low regardless of the ENSO phase
```

# Checking Scanner Types

- So far, we have assumed the user is going to give the right information to the Scanner object,
  - e.g. winter or summer for the season
  - e.g. a number for the index
- What happens if they give something we don't expect?
  - E.g. what if they provide spring for the season?
  - E.g. what if they provide "EL Nino" for the index?
- Issues: a bug in the program or worse, a **run time error!**
- **Solution: Use if statements to check the scanner types**

# Checking with Strings Input

- Recall: The Scanner object uses the method `.nextLine()` to read in the next input as a string
- If you are expecting a string, you can make sure the input is permissible using an `if` statement
- Example:

```
String season = in.nextLine()
if (!season.equals("winter"))
{
    System.out.println("Season is not winter");
    return;
}
```

The statement `return` will end the `main()` method abruptly!

# Checking for Ints or Doubles

- We have seen that the Scanner class has methods that expect the inputs to be of a certain type:
  - `nextDouble()`
  - `nextInt()`
- We can also check the inputs (before reading them into memory!) that the inputs are the types we are expecting
  - `hasNextDouble()`
  - `hasNextInt()`
- These methods return `true` if the type is as expect; `false` otherwise
- Be sure to call these methods BEFORE you read in the input to memory

# Example: NestedENSOWithCheck.java

Sample Outputs #1-3, same as before

Sample Output #4    Enter a season (winter or summer): **spring**

Invalid season: spring

Sample Output #5    Enter a season (winter or summer): **summer**

Enter the ENSO index: **El Nino**

Invalid index: El Nino

Using Scanner methods and if statements, we can check  
the user inputs to ensure they are the types we expect

# Participation Exercise 5-2a: BooleanColors

Goal: Write a program that reads in two different colors and prints some messages depending on which colors are provided

Sample output #1

```
Enter your first favorite color: green
Enter your second favorite color: blue
Those colors are my favorite too!
Neither of the colors is a "warm" color.
```

Sample output #2

```
Enter your first favorite color: RED
Enter your second favorite color: yellow
Those colors are ok, I guess...
One of the colors is a "warm" color.
```

Sample output #3

```
Enter your first favorite color: green
Enter your second favorite color: purple
Those colors are ok, I guess...
Neither of the colors is a "warm" color.
```

Codecheck Link: [HERE](#) and on Canvas

# Participation Exercise 5-2b: IceCreamShop

Goal: Write a program that provides different output based on user requests for different combinations of cones and scoops at an ice cream shop

## Sample output #1

```
Enter 1 for Plain or 2 for Waffle: 1
How many scoops (1 or 2): 1
Your payment is $1.75.
```

## Sample output #2

```
Enter 1 for Plain or 2 for Waffle: 2
How many scoops (1 or 2): 2
Your payment is $2.75.
```

## Sample output #3

```
Enter 1 for Plain or 2 for Waffle: 3
Invalid type: 3
Program terminated.
```

## Sample output #4

```
Enter 1 for Plain or 2 for Waffle: 2
How many scoops (1 or 2): 3
Invalid scoops: 3
Program terminated.
```

Codecheck Link: [HERE](#) and on Canvas