Java Programming

CPT111 – Lecture 4

Erick Purwanto

Xi'an Jiaotong-Liverpool University

西交利物浦大學

CPT111 Java Programming

Lecture 4

Flow Controls

Welcome!

- Welcome to Lecture 4 – Flow Controls

- In this lecture we are going to learn about
  - boolean
    - boolean and comparison operators
  - conditionals
    - `if`, `if-else` statements
    - nested `if`s
  - loops
    - `while` loops
    - infinite loops
    - logical error – off-by-one bug

Part 1: Boolean

- In the first part of the lecture, we are going to discuss another data type that is frequently used called boolean
  - from mathematician George Boole
  - Boolean algebra

Recall in Week 3: Boolean

- Boolean type only has two values: `true` or `false`
  - useful for checking if certain conditions are met to control program flow

```
public class BooleanValues {
    public static void main(String[] args) {
        boolean a = true;
        boolean b = false;
        System.out.println(a);
        System.out.println(b);
    }
}
```

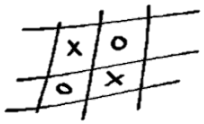
Boolean Operators

- Operators defined for boolean:
  - **and**: a && b is true if both a and b are true, and false otherwise
  - **or**: a || b is true if either a or b is true (or both are true), and false otherwise
  - **not**: !a is true if a is false, and false otherwise

a	!a
true	
false	

a	b	a && b	a || b
true	true		
true	false		
false	true		
false	false		

In-Class Quiz 1



- How can we implement **XOR** using boolean operators **and**, **or**, **not** ?
  - `(!a && b) || ( a && !b)`
  - `( a && b) || (!a && !b)`
  - `(!a || b) && ( a || !b)`
  - `( a || b) && (!a || !b)`

Comparison Operators

- Comparison operators that take operands of one type (e.g., `int` or `double`) and produce a result of type `boolean`

operator	meaning	`true`	`false`
==	equal		
!=	not equal		
<	less than		
<=	less than or equal		
>	greater than		
>=	greater than or equal		

  - do not confuse equality == with assignment operator =

In-Class Quiz 2

- Given an integer variable month, give an expression of legal month:
  - ❑ `(month >  1) && (month <  12)`
  - ❑ `(month >  1) || (month <  12)`
  - ❑ `(month >= 1) && (month <= 12)`
  - ❑ `(month >= 1) || (month <= 12)`
  - ❑ `(month >  0) && (month <  13)`
  - ❑ `(month >  0) || (month <  13)`

Precedence

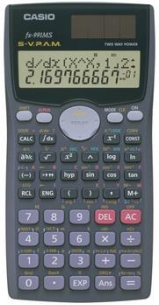
- Operator precedence / order of execution:

Operators	Precedence
postfix	*expr*++ *expr*--
unary	! ++*expr* --*expr*
multiplicative	* / %
additive	+ -
relational	< > <= >=
equality	== !=
logical and	&&
logical or	||
assignment	= += -= *= /= %=

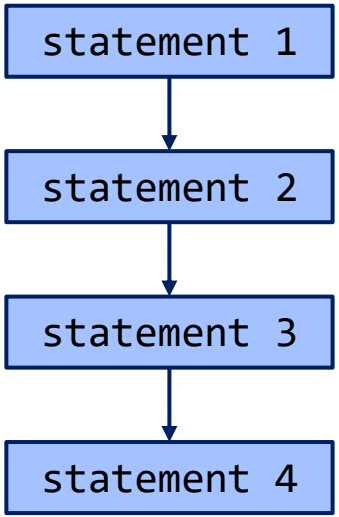
Part 2: Conditionals

- In the second part of the lecture, we will discuss conditionals
  - achieved by the `if` statements

Recall in Week 2 and Week 3

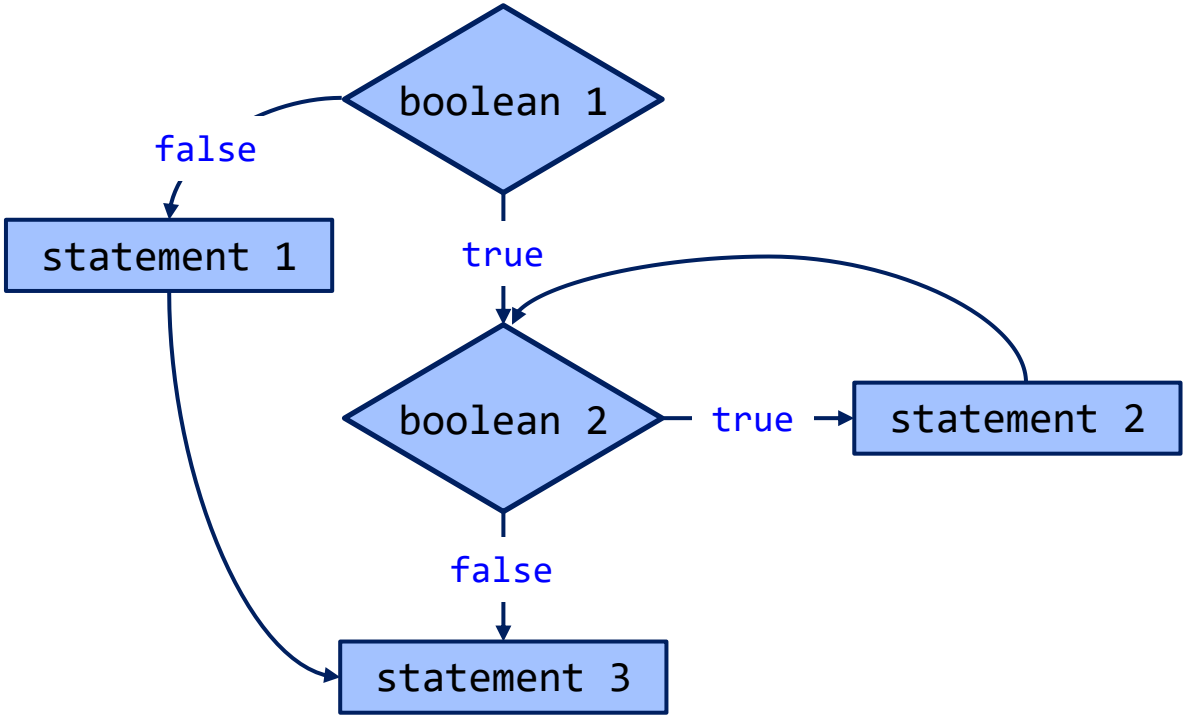


- Week 2: calculate, display
- Week 3: get input, calculate and display
  - equivalent of a calculator
  - straight-line control flow



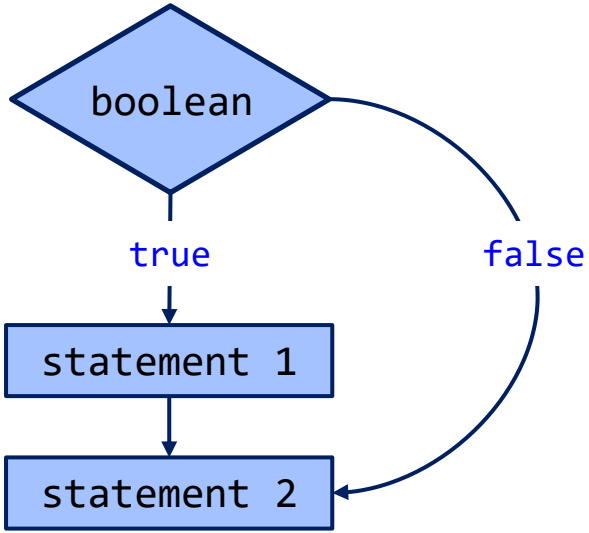
Flow Controls

- Conditionals and loops enables us to control the flow of the program



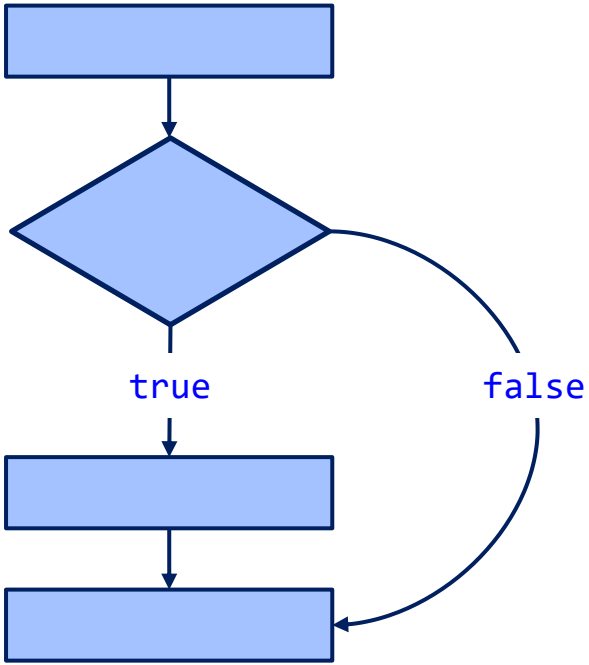
`if` statement

- `if` statement executes `if`-block (statements inside curly brackets following `if`) depending on the values of variables in boolean expression
  - evaluate the boolean expression
  - if true, execute `if`-block
  - if false, do nothing, go to next statement



`if` statement example

- Write a Java program that compute the absolute value of user integer input



`if` statement example

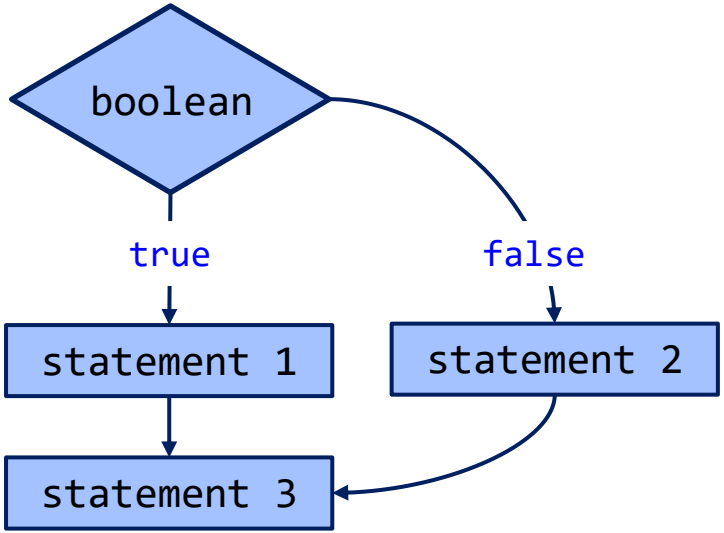
- Write a Java program that compute the absolute value of user integer input

```
public class AbsoluteIntValue {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input;
        int num;
        input = scanner.nextLine();
        num = Integer.parseInt(input);

        if (num < 0) {
            num = -num;
        }
        System.out.println(num);
    }
}
```

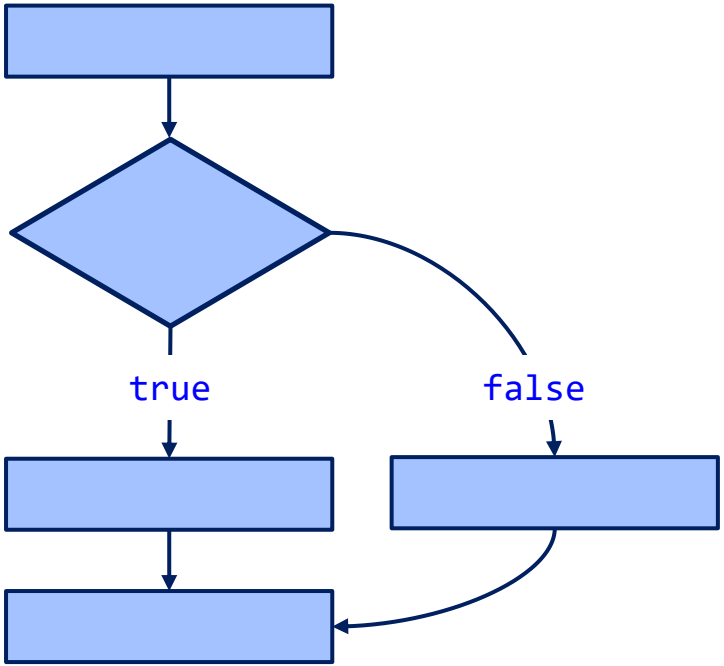
`if-else` statement

- `if-else` statement executes `if`-block (statements inside curly brackets following `if`) depending on the values of variables in boolean expression
  - evaluate the boolean expression
  - if true, execute `if`-block
  - if false, execute `else`-block



`if-else` statement example

- Write a Java program that compute the maximum of two values of user integer input



`if-else` statement example

- Write a Java program that compute the maximum of two values of user integer input

```
public class MaximumIntValue {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input;
        int x, y;
        input = scanner.nextLine();
        x = Integer.parseInt(input);
        input = scanner.nextLine();
        y = Integer.parseInt(input);

        int max;
        if (x > y) {
            max = x;
        }
        else {
            max = y;
        }
        System.out.println(max);
    }
}
```

In-Class Quiz 3

- Which of the following will lead to **a compile error**?
  Assume a, b, c are already initialized.

  - ❑ `if a > b { c = 5; }`
  - ❑ `if (a > b) ( c = 5; )`
  - ❑ `if (a > b) then c = 5;`
  - ❑ `if (a > b) c = 5 else c = 2;`

Multiple `if-else` statements

- Write a Java program that report whether a user integer input is "Negative", "Zero", or "Positive"

```
public class NegZeroPos {
    public static void main(String[] args) {

        // code to receive user integer input to num

        if (num < 0) {
            System.out.println("Negative");
        }
        else if (num == 0) {
            System.out.println("Zero");
        }
        else {
            System.out.println("Positive");
        }
    }
}
```

Multiple `if`s and logical operators

- Write a Java program to print the Degree Class given user's Percentage Score according to British marking criteria

Percentage Score	Degree Class
70% - 100%	First Class
60% - 69%	Upper Second Class
50% - 59%	Lower Second Class
40% - 49%	Third Class
0% - 39%	Fail

```
public class BritishMarkingCriteria {
    public static void main(String[] args) {

        // code to receive user double input to score

        if (score >= 70)
            System.out.println("First Class");
        if (score >= 60 && score < 70)
            System.out.println("Upper Second Class");
        if (score >= 50 && score < 60)
            System.out.println("Lower Second Class");
        if (score >= 40 && score < 50)
            System.out.println("Third Class");
        if (score < 40)
            System.out.println("Fail");
    }
}
```

Nested `if-else`s

- Write a Java program to print the Degree Class given user's Percentage Score according to British marking criteria

Percentage Score	Degree Class
70% - 100%	First Class
60% - 69%	Upper Second Class
50% - 59%	Lower Second Class
40% - 49%	Third Class
0% - 39%	Fail

```
...
if (score < 40)
    System.out.println("Fail");
else {
    if (score < 50)
        System.out.println("Third Class");
    else {
        if (score < 60)
            System.out.println("Lower Second Class");
        else {
            if (score < 70)
                System.out.println("Upper Second Class");
            else
                System.out.println("First Class");
        }
    }
}
```

Part 3: Loops

- In the third part of the lecture we are going to discuss loops
  - in particular, `while` loops
  - the other type of loops, the `for` loops, will be discussed next week

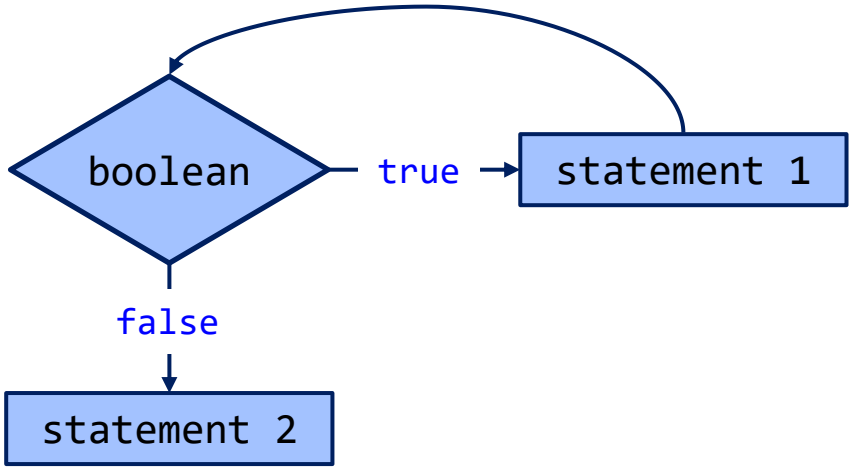
Loops

- Previously we introduced flow control
  - to branch out the execution of a program
- What if we want to execute a statement multiple times?
  - many computations are inherently repetitive
  - for example, count down from 10 to 1

```
public class CountDown {
    public static void main(String[] args) {
        System.out.println("10");
        System.out.println("9");
        ...
        System.out.println("1");
    }
}
```

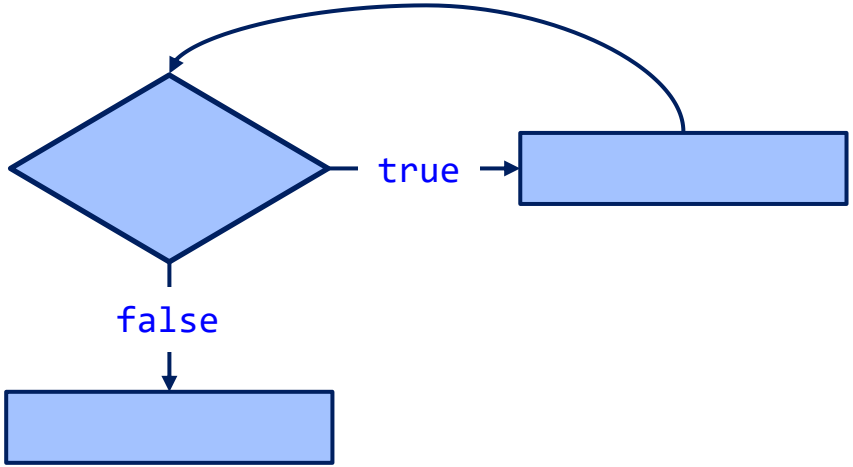
`while` loops

- `while` loop executes statements repeatedly when certain conditions are met
  - evaluate the boolean expression
  - if true, execute some statements
  - repeat



`while` loops example

- Write a Java program that print count down from 10 to 1 using `while` loops



`while` loops example

- Write a Java program that print count down from 10 to 1 using `while` loops

```
public class CountDown {
    public static void main(String[] args) {
        int count = 10;
        while (count >= 1) {
            System.out.println(count);
            count = count - 1;
        }
    }
}
```

initialization – codes to give initial value to variable in the boolean expression

test – boolean expression that must be `true` before each iteration

work – codes to be repeated for each iteration

update – code to make the boolean expression closer to be `false`

`while` loops example

- Write a Java program that print count down from 10 to 1 using `while` loops

```
public class CountDown {
    public static void main(String[] args) {
        int count = 10;
        while (count >= 1) {
            System.out.println(count);
            count = count - 1;
        }
    }
}
```

what would happen
if we initialize `count` to 0 ?

`while` loops example

- Write a Java program that print count down from 10 to 1 using `while` loops

```
public class CountDown {
    public static void main(String[] args) {
        int count = 10;
        while (count >= 1) {
            System.out.println(count);
            count = count - 1;
        }
    }
}
```

what would happen
if we forgot to update `count`?
or, if we increment instead of decrement?

Infinite loop

- Program never stops looping
  - forget to update stopping condition
  - or, stopping condition is never met

```java
public class Halloween {
    public static void main(String[] args) {

        while (true) {
            System.out.println("Halloween!");
        }
    }
}
```

Powers of Two

- Write a Java program that takes integer input n and prints n first powers of two starting from $2^0 = 1$

```
public class PowersOfTwo {
    public static void main(String[] args) {

        // code to receive user integer input to n

        int i = 1, val = 1;
        while (i <= n) {
            System.out.println(val);
            val = val * 2;
            i = i + 1;
        }
    }
}
```

Off-by-one Bug

- Write a Java program that takes integer input n and prints n first powers of two starting from $2^0 = 1$

```
public class PowersOfTwo {
    public static void main(String[] args) {

        // code to receive user integer input to n

        int i = 1, val = 1;
        while (i < n) {
            System.out.println(val);
            val = val * 2;
            i = i + 1;
        }
    }
}
```

if we use < instead of <= we will miss to print one powers of two

this is a common **logical bug** in computer programming known as **off-by-one bug**

In-Class Quiz 4

- What is wrong with this Java program that is supposed to print n first powers of two ?

```
public class PowersOfTwo {
    public static void main(String[] args) {

        // code to receive user integer input to n

        int i = 1, val = 1;
        while (i <= n)
            System.out.println(val);
            val = val * 2;
            i = i + 1;
    }
}
```

- compile error
- infinite loop
- off-by-one bug
- nothing's wrong, print correct output

Powers of Two 2

- `while` loops is especially useful when you ***don't exactly know how many times needed to loop over***

- Write a Java program that takes integer input n and prints all powers of two starting from $2^0 = 1$ that is less than n
  - discussed in Lab 4

Thank you for your attention !

- In this lecture, you have learned:
  - to use comparison and boolean operators
  - to branch out using `if`, `if-else` and nested `if`s statements
  - to compute repeatedly using `while` loops
  - to watch out for infinite loops and off-by-one bugs

- Please continue to Lab 4, and solve
  - Exercise #4.1, #4.2 and
  - CW1 #4.1, #4.2