# Chapter Six: More Conditionals and Loops

## CTEC 150, Fall 2019

Jay Jain

Bossier Parish Community College

*jjain@bpcc.edu*

September 2019

# 6.1: The `switch` Statement

- **6.1: The `switch` Statement**
- 6.2: The Conditional Operator
- 6.3: The `do` Statement
- 6.4: The `for` Statement

# The `switch` Statement

- Evaluates an expression and attempts to match the result to multiple *cases*

# The `switch` Statement

- Evaluates an expression and attempts to match the result to multiple *cases*
- Flow of control transfers to statement associated with first case value that matches the expression

# The `switch` Statement

- Evaluates an expression and attempts to match the result to multiple *cases*
- Flow of control transfers to statement associated with first case value that matches the expression
- `switch` and `case` are reserved words

## Syntax

```
switch ( expression ){
case value1 :
    statement1
case value2 :
    statement2
case value3 :
    statement3
default :
    statement4
}
```

# The `switch` Statement

- A `break` statement is used as the last statement in each case's statement list

# The `switch` Statement

- A `break` statement is used as the last statement in each case's statement list
- The `break` statement transfers control to the end of the `switch` statement

# The `switch` Statement

- A `break` statement is used as the last statement in each case's statement list
- The `break` statement transfers control to the end of the `switch` statement
- The `default` case has no associated value and used when no other case value matches

# The `switch` Statement

- A `break` statement is used as the last statement in each case's statement list
- The `break` statement transfers control to the end of the `switch` statement
- The `default` case has no associated value and used when no other case value matches
- The type of a `switch` expression can only be integer, character, enumerated type, or String type. (No floating points)

# switch statement example

## Example

```
switch (option){
    case 'A':
        aCount++;
        break;
    case 'B':
        bCount++;
        break;
    case 'C':
        cCount++;
        break;
    default:
        System.out.println("If no cases match...");
}
```

# 6.2: The Conditional Operator

- 6.1: The `switch` Statement
- **6.2: The Conditional Operator**
- 6.3: The `do` Statement
- 6.4: The `for` Statement

# The Conditional Operator

- The conditional operator evaluates to one of two expressions based on a boolean expression (another way to write an `if` statement)

**Syntax**

```
condition ? expression1 : expression2
```

# The Conditional Operator

- The conditional operator evaluates to one of two expressions based on a boolean expression (another way to write an `if` statement)
- If the condition is true, expression1 is evaluated

**Syntax**

```
condition ? expression1 : expression2
```

# The Conditional Operator

- The conditional operator evaluates to one of two expressions based on a boolean expression (another way to write an `if` statement)
- If the condition is true, expression1 is evaluated
- If the condition is false, expression2 is evaluated

**Syntax**

```
condition ? expression1 : expression2
```

# The Conditional Operator

## Example 1

```
larger = ((num1 > num2) ? num1 : num2);
```

## Example 2

```
System.out.println("Your change is " + count +
   ((count == 1) ? "Dime" : "Dimes"));
```

## Example 3

```
if (val <= 10)
        System.out.println("It is not greater than 10.");
else
        System.out.println("It is greater than 10.");

System.out.println("It is" +
      ((val <= 10) ? " not" : "") +
      " greater than 10.");
```

# 6.3: The do Statement

- 6.1: The switch Statement
- 6.2: The Conditional Operator
- **6.3: The do Statement**
- 6.4: The for Statement

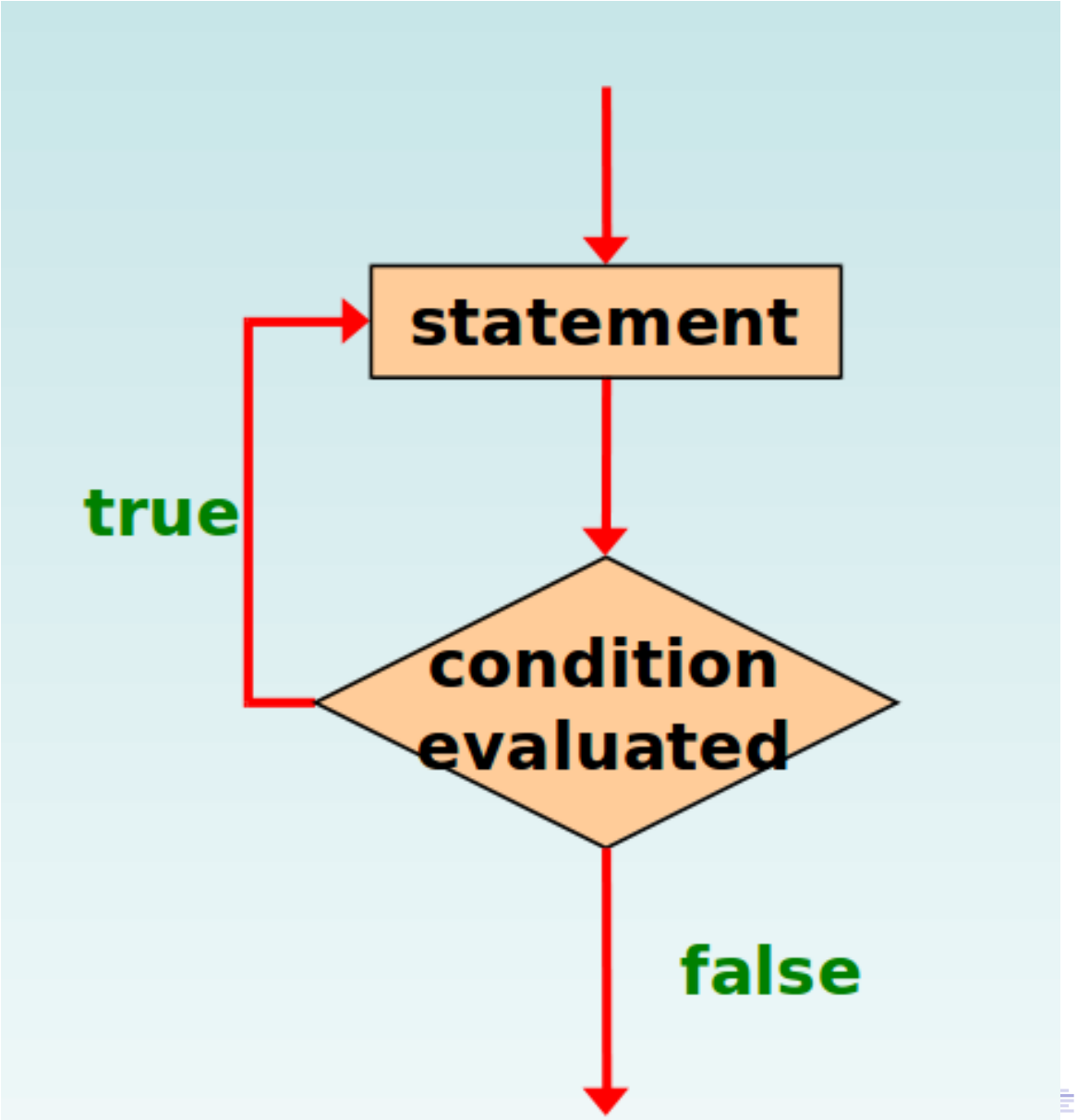
# The do Statement

- The statement-list is executed once initially, **and then** the condition is evaluated
- The statement is executed repeatedely until the condition becomes false

## Syntax

```
do{
    statement-list;
}while (condition);
```

# The do Statement

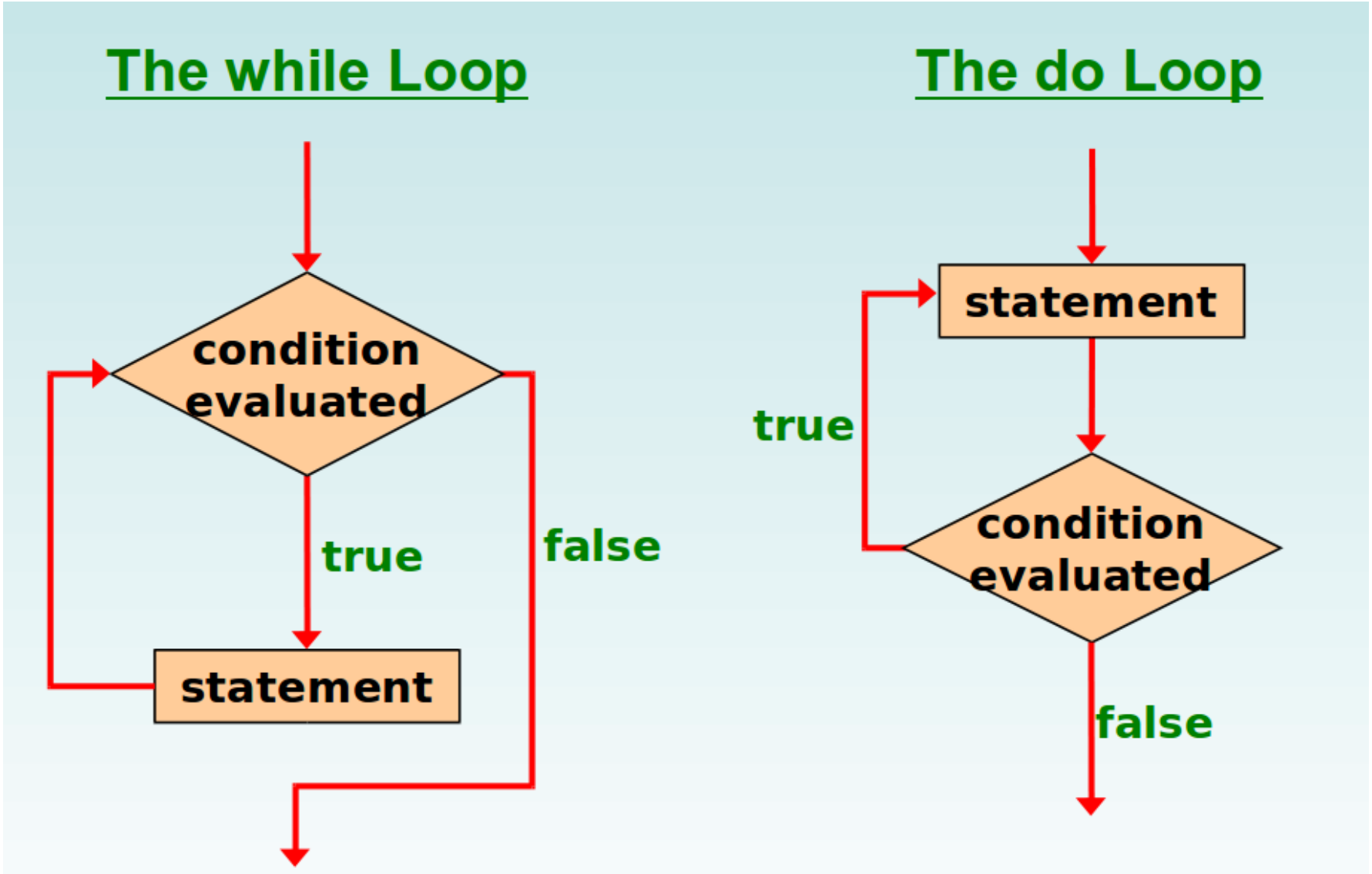


# The do Statement

## Example 1

```
int count = 0;
do{
    count++;
    System.out.println(count);
} while (count < 5);
```

# Comparing `while/do` Statements



# 6.4: The `for` Statement

- 6.1: The `switch` Statement
- 6.2: The Conditional Operator
- 6.3: The `do` Statement
- **6.4: The `for` Statement**

# The `for` Statement

- The initialization is executed once before the loop begins

**Syntax**

```
for ( initialization ; condition ; increment){
    statement;
}
```

# The `for` Statement

- The initialization is executed once before the loop begins
- The statement is executed until the condition becomes false

**Syntax**

```
for ( initialization ; condition ; increment){
    statement;
}
```

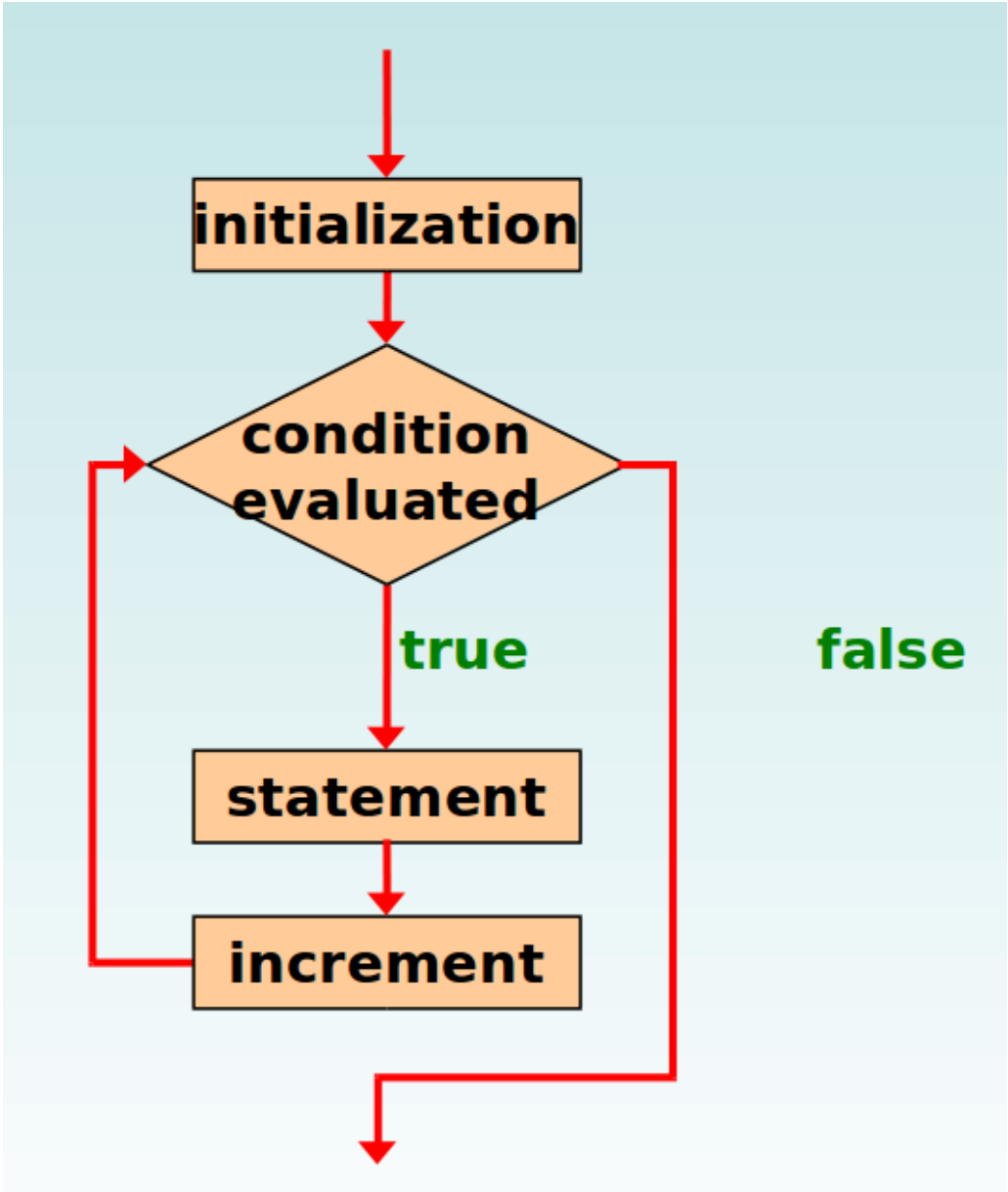
# The `for` Statement

- The initialization is executed once before the loop begins
- The statement is executed until the condition becomes false
- The increment portion is executed at the end of each iteration

**Syntax**

```
for ( initialization ; condition ; increment){
    statement;
}
```

# The `for` Statement



# The for Statement

## Equivalent while Statement

```
initialization;
while ( condition )
{
        statement;
        increment;
}
```

# The for Statement

## Equivalent while Statement

```
initialization;
while ( condition )
{
        statement;
        increment;
}
```

## Example 1

```
for (int count = 1; count <= 5; count++)
   System.out.println(count);
```

# The for Statement

## Equivalent while Statement

```
initialization;
while ( condition )
{
        statement;
        increment;
}
```

## Example 1

```
for (int count = 1; count <= 5; count++)
   System.out.println(count);
```

## Example 2

```
for (int num = 100; num > 0; num -= 5)
   System.out.println(num);
```

# QUESTIONS ???

*You don't understand anything until you learn it more than one way.*

*- Marvin Minsky*