CONDITIONAL OPERATORS

Introduction to Conditional Operators

Java includes operators that only use boolean values.

These *conditional operators* help simplify expressions containing complex boolean relationships by reducing multiple boolean values to a single value: `true` or `false`.

For example, what if we want to run a code block only if *multiple* conditions are true. We could use the *AND* operator: `&&`.

A	B	A && B
True	True	True
True	False	False
False	True	False
False	False	False

Or, we want to run a code block if *at least one* of two conditions are `true`. We could use the *OR* operator: `||`.

A	B	A || B
True	True	True
True	False	True
False	True	True
False	False	False

Finally, we can produce the opposite value, where `true` becomes `false` and `false` becomes `true`, with the *NOT* operator: `!`.

A	!A
True	False
False	True

Understanding these complex relationships can feel overwhelming at first. Luckily, *truth tables*, like the ones seen to the right, can assist us in determining the relationship between two boolean-based conditions.

In this lesson, we'll explore each of these conditional operators to see how they can be implemented into our conditional statements.

Instructions

The text editor contains a `Reservation` class we'll build in this lesson.

Note the different conditional statements and operators that we're using to control the execution of the program.

Move on when you're ready!

```
public class Reservation {
  int guestCount;
  int restaurantCapacity;
  boolean isRestaurantOpen;
  boolean isConfirmed;

  public Reservation(int count, int capacity, boolean open) {
    if (count < 1 || count > 8) {
      System.out.println("Invalid reservation!");
    }
    guestCount = count;
    restaurantCapacity = capacity;
    isRestaurantOpen = open;
  }

  public void confirmReservation() {
    if (restaurantCapacity >= guestCount && isRestaurantOpen) {
      System.out.println("Reservation confirmed");
```

```java
      isConfirmed = true;
    } else {
      System.out.println("Reservation denied");
      isConfirmed = false;
    }
  }

  public void informUser() {
    if (!isConfirmed) {
      System.out.println("Unable to confirm reservation, please 
contact restaurant.");
    } else {
      System.out.println("Please enjoy your meal!");
    }
  }

  public static void main(String[] args) {
    Reservation partyOfThree = new Reservation(3, 12, true);
    Reservation partyOfFour = new Reservation(4, 3, true);
    partyOfThree.confirmReservation();
    partyOfThree.informUser();
    partyOfFour.confirmReservation();
    partyOfFour.informUser();
  }
}
```

```
Reservation confirmed
Please enjoy your meal!
Reservation denied
Unable to confirm reservation, please contact restaurant.
```

Conditional-And: &&

Let's return to our student enrollment program. We've added an additional requirement: not only must students have the prerequisite, but their tuition must be paid up as well. We have *two* conditions that must be `true` before we enroll the student.

Here's one way we could write the code:

```
if (tuitionPaid) {
  if (hasPrerequisite) {
    // enroll student
  }
}
```

We've nested two `if-then` statements. This does the job but we can be more concise with the *AND* operator:

```
if (tuitionPaid && hasPrerequisite) {
  // enroll student
}
```

The AND operator, `&&`, is used between two boolean values and evaluates to a single boolean value. If the values **on both sides** are `true`, then the resulting value is `true`, otherwise the resulting value is `false`.

This code illustrates every combination:

```
true && true
// true
false && true
// false
true && false
// false
false && false
// false
```

Instructions

1.Our Reservation class has the method confirmReservation() which validates if a restaurant can accomodate a given reservation.
We need to build the conditional logic into confirmReservation() using three instance variables:

- restaurantCapacity
- guestCount
- isRestaurantOpen

Use an if-then-else statement:

If restaurantCapacity is greater than or equal to guestCount **and** the restaurant is open, print "Reservation confirmed" and set isConfirmed to true.

else print "Reservation denied" and set isConfirmed to false.

Note: For now, the informUser() method will always print "Please enjoy your meal" even if the reservation was not confirmed. We will modify this method in an upcoming exercise!

```
public class Reservation {
  int guestCount;
  int restaurantCapacity;
  boolean isRestaurantOpen;
  boolean isConfirmed;

  public Reservation(int count, int capacity, boolean open) {
    guestCount = count;
    restaurantCapacity = capacity;
    isRestaurantOpen = open;
  }

  public void confirmReservation() {
    /*
```

```java
    Write conditional
    ~~~~~~~~~~~~~~~~~~~
    if restaurantCapacity is greater
    or equal to guestCount
    AND
    the restaurant is open:
      print "Reservation confirmed"
      set isConfirmed to true
    else:
      print "Reservation denied"
      set isConfirmed to false
  */
  if (restaurantCapacity >= guestCount && isRestaurantOpen) {
        System.out.println("Reservation Confirmed.");
        isConfirmed = true;
}else {
   System.out.println("Reservation denied.");
    isConfirmed = false;
}
}

public void informUser() {
  System.out.println("Please enjoy your meal!");
}

public static void main(String[] args) {
  Reservation partyOfThree = new Reservation(3, 12, true);
  Reservation partyOfFour = new Reservation(4, 3, true);
  partyOfThree.confirmReservation();
  partyOfThree.informUser();
  partyOfFour.confirmReservation();
  partyOfFour.informUser();
}
}
```

```
Output:
Reservation Confirmed.
Please enjoy your meal!
Reservation denied.
Please enjoy your meal!
```

Conditional-Or: ||

The requirements of our enrollment program have changed again. Certain courses have prerequisites that are satisfied by multiple courses. As long as students have taken **at least one** prerequisite, they should be allowed to enroll.

Here's one way we could write the code:

```
if (hasAlgebraPrerequisite) {
  // Enroll in course
}
if (hasGeometryPrerequisite) {
  // Enroll in course
}
```

We're using two different `if-then` statements with **the same code block**. We can be more concise with the *OR* operator:

```
if (hasAlgebraPrerequisite || hasGeometryPrerequisite) {
  // Enroll in course
}
```

The OR operator, `||`, is used between two boolean values and evaluates to a single boolean value. If **either of the two values** is `true`, then the resulting value is `true`, otherwise the resulting value is `false`.

This code illustrates every combination:

```
true || true
// true
false || true
// true
true || false
// true
false || false
// false
```

Keep Reading: AP Computer Science A Students

On some occasions, the compiler can determine the truth value of a logical expression by only evaluating the first boolean operand; this is known as *short-circuited evaluation*. Short-circuited evaluation only works with expressions that use && or ||.

In an expression that uses ||, the resulting value will be true as long as one of the operands has a true value. If the first operand of an expression is true, we don't need to see what the value of the other operand is to know that the final value will also be true.

For example, we can run the following code without error despite dividing a number by 0 in the second operand because the first operand had a true value:

```
if (1 > 0 || 2 / 0 == 7) {
  System.out.println("No errors here!");
}
```

An expression that uses && will only result in true if both operands are true. If the first operand in the expression is false, the entire value will be false.

Instructions

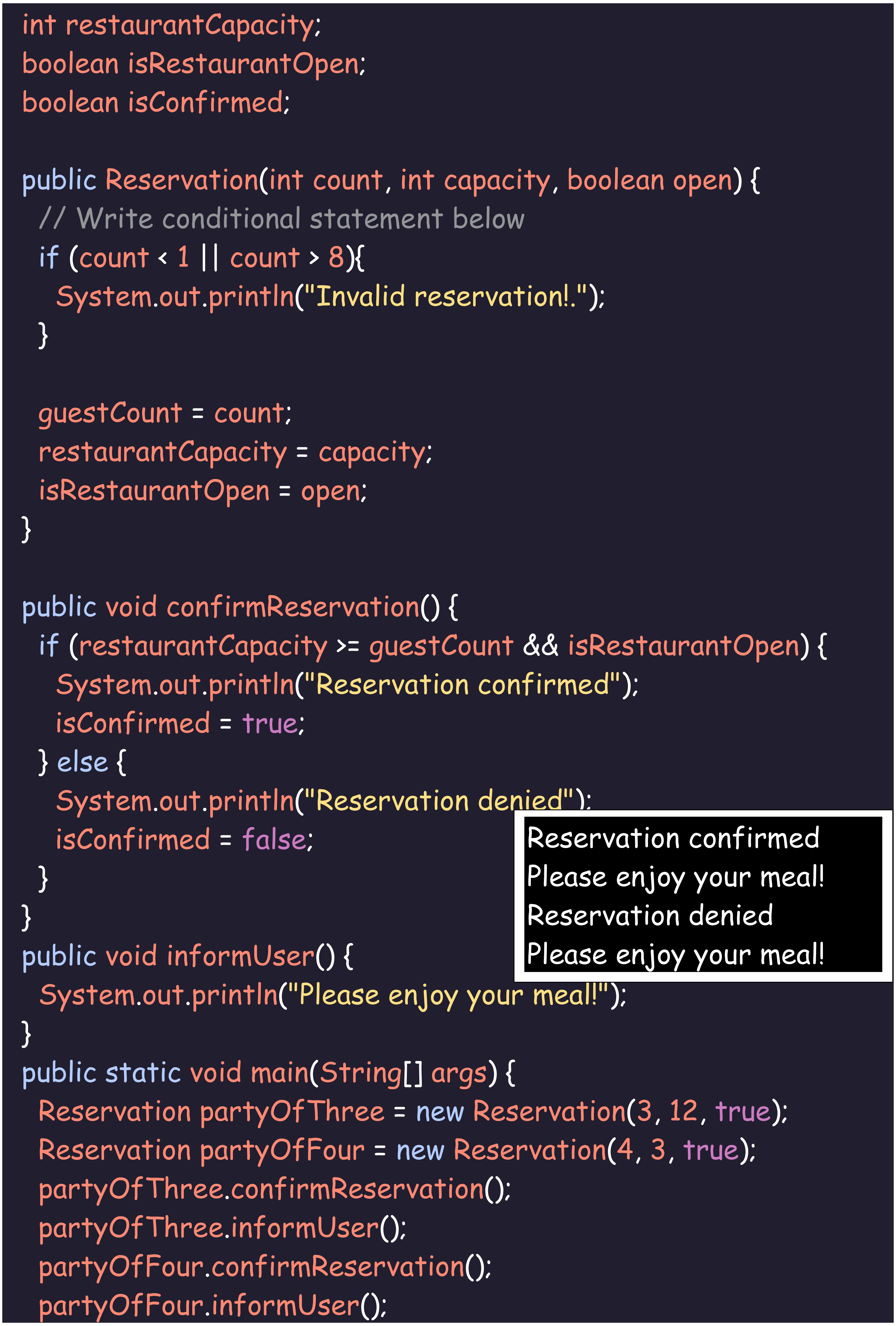
1.Let's write a message inside the Reservation() constructor that warns against bad input.

Our restaurants can't seat parties of more than 8 people, and we don't want reservations for 0 or less because that would be silly.

Inside Reservation(), write a conditional that uses ||.

If count is less than 1 **OR** greater than 8 we want to write the following message: Invalid reservation!.

```
public class Reservation {
  int guestCount;
```



```java
int restaurantCapacity;
boolean isRestaurantOpen;
boolean isConfirmed;

public Reservation(int count, int capacity, boolean open) {
  // Write conditional statement below
  if (count < 1 || count > 8){
    System.out.println("Invalid reservation!.");
  }

  guestCount = count;
  restaurantCapacity = capacity;
  isRestaurantOpen = open;
}

public void confirmReservation() {
  if (restaurantCapacity >= guestCount && isRestaurantOpen) {
    System.out.println("Reservation confirmed");
    isConfirmed = true;
  } else {
    System.out.println("Reservation denied");
    isConfirmed = false;
  }
}
public void informUser() {
  System.out.println("Please enjoy your meal!");
}
public static void main(String[] args) {
  Reservation partyOfThree = new Reservation(3, 12, true);
  Reservation partyOfFour = new Reservation(4, 3, true);
  partyOfThree.confirmReservation();
  partyOfThree.informUser();
  partyOfFour.confirmReservation();
  partyOfFour.informUser();
```

```
Reservation confirmed
Please enjoy your meal!
Reservation denied
Please enjoy your meal!
```

Logical NOT: !

The *unary* operator NOT, !, works on a **single** value. This operator evaluates to the opposite boolean to which it's applied:

```
!false
// true
!true
// false
```

NOT is useful for expressing our intent clearly in programs. For example, sometimes we need the opposite behavior of an if-then: run a code block **only if** the condition is false.

```
boolean hasPrerequisite = false;

if (hasPrerequisite) {
  // do nothing
} else {
  System.out.println("Must complete prerequisite course!");
}
```

This code does what we want but it's strange to have a code block that does nothing!

The logical NOT operator cleans up our example:

```
boolean hasPrerequisite = false;

if (!hasPrerequisite) {
  System.out.println("Must complete prerequisite course!");
}
```

We can write a succinct conditional statement without an empty code block.

Instructions

1.Let's make informUser() more informative. If their reservation is not confirmed, they should know!

Write an if-then-else statement and use ! with isConfirmed as the

condition.
If their reservation is **not** confirmed, write `Unable to confirm reservation, please contact restaurant.`

Else write: `Please enjoy your meal!`

```java
public class Reservation {
  int guestCount;
  int restaurantCapacity;
  boolean isRestaurantOpen;
  boolean isConfirmed;

  public Reservation(int count, int capacity, boolean open) {
    if (count < 1 || count > 8) {
      System.out.println("Invalid reservation!");
    }
    guestCount = count;
    restaurantCapacity = capacity;
    isRestaurantOpen = open;
  }

  public void confirmReservation() {
    if (restaurantCapacity >= guestCount && isRestaurantOpen) {
      System.out.println("Reservation confirmed");
      isConfirmed = true;
    } else {
      System.out.println("Reservation denied");
      isConfirmed = false;
    }
  }

  public void informUser() {
    // Write conditional here
if(!isConfirmed){
  System.out.println("Unable to confirm reservation, please contact
restaurant.");
}else {
  System.out.println("Please enjoy your meal!");
}
  }

  public static void main(String[] args) {
    Reservation partyOfThree = new Reservation(3, 12, true);
    Reservation partyOfFour = new Reservation(4, 3, true);
    partyOfThree.confirmReservation();
    partyOfThree.informUser();
    partyOfFour.confirmReservation();
    partyOfFour.informUser();
  }
}
```

```
Reservation confirmed
Please enjoy your meal!
Reservation denied
Unable to confirm reservation, please contact restaurant
```

Combining Conditional Operators

We have the ability to expand our boolean expressions by using multiple conditional operators in a single expression.

For example:

```
boolean foo = true && !(false || !true)
```

How does an expression like this get evaluated by the compiler? The order of evaluation when it comes to conditional operators is as follows:

1. Conditions placed in parentheses - `()`
2. NOT - `!`
3. AND - `&&`
4. OR - `||`

Using this information, let's dissect the expression above to find the value of `foo`:

```
true && !(false || !true)
```

First, we'll evaluate `(false || !true)` because it is enclosed within parentheses. Following the order of evaluation, we will evaluate `!true`, which equals `false`:

```
true && !(false || false)
```

Then, we'll evaluate `(false || false)` which equals `false`. Now our expression looks like this:

```
true && !false
```

Next, we'll evaluate `!false` because it uses the NOT operator. This expression equals `true` making our expression the following:

```
true && true
```

true && true evaluates to true; therefore, the value of foo is true.

Instructions
Take a look at the three expressions in **Operators.java**.

Using your understanding of the order of execution, find out whether the value of each expression is true or false.

When you're ready, uncomment the print statements to find out if you are right.

```
public class Operators {
  public static void main(String[] args) {
    int a = 6;
    int b = 3;

    boolean ex1 = !(a == 7 && (b >= a || a != a));
    System.out.println(ex1);

    boolean ex2 = a == b || !(b > 3);
    System.out.println(ex2);

    boolean ex3 = !(b <= a && b != a + b);
     System.out.println(ex3);

  }
}
```

```
true
true
false
```

Review

Conditional operators work on boolean values to simplify our code. They're often combined with conditional statements to consolidate the branching logic.

Conditional-AND, `&&`, evaluates to `true` if the booleans on *both sides* are `true`.

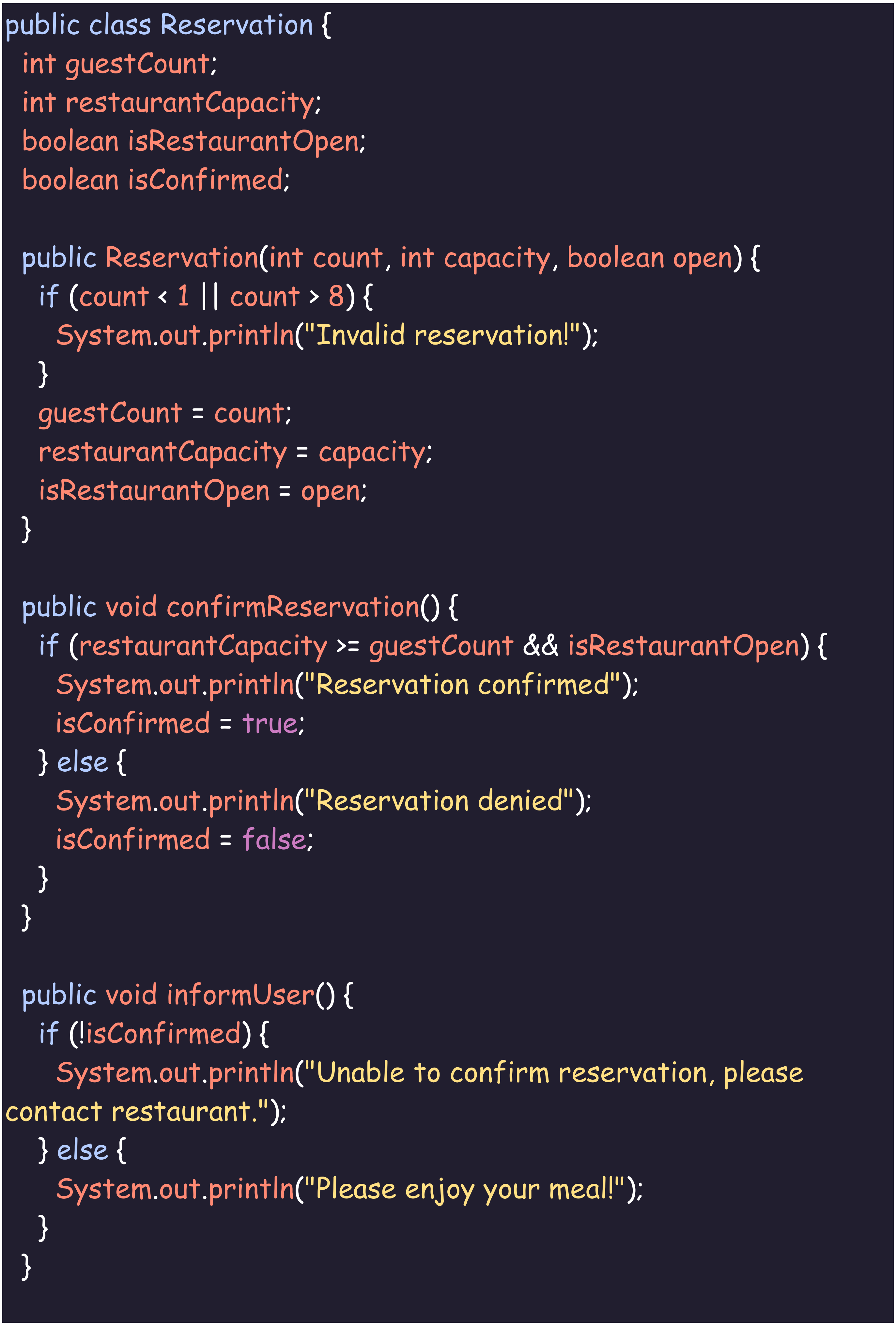
```
if (true && false) {
  System.out.println("You won't see me print!");
} else if (true && true) {
  System.out.println("You will see me print!");
}
```

Conditional-OR, `||`, evaluates to `true` if one or both of the booleans on either side is `true`.

```
if (false || false) {
  System.out.println("You won't see me print!");
} else if (false || true) {
  System.out.println("You will see me print!");
}
```

Logical-NOT, `!`, evaluates to the opposite boolean value to which it is applied.

```
if (!false) {
  System.out.println("You will see me print!");
}
```



```java
public class Reservation {
  int guestCount;
  int restaurantCapacity;
  boolean isRestaurantOpen;
  boolean isConfirmed;

  public Reservation(int count, int capacity, boolean open) {
    if (count < 1 || count > 8) {
      System.out.println("Invalid reservation!");
    }
    guestCount = count;
    restaurantCapacity = capacity;
    isRestaurantOpen = open;
  }

  public void confirmReservation() {
    if (restaurantCapacity >= guestCount && isRestaurantOpen) {
      System.out.println("Reservation confirmed");
      isConfirmed = true;
    } else {
      System.out.println("Reservation denied");
      isConfirmed = false;
    }
  }

  public void informUser() {
    if (!isConfirmed) {
      System.out.println("Unable to confirm reservation, please
contact restaurant.");
    } else {
      System.out.println("Please enjoy your meal!");
    }
  }
```

```
 public static void main(String[] args) {
   // Create i
   Reservation groupOf3 = new Reservation(3, 12, true);
       groupOf3.confirmReservation();
       groupOf3.informUser();

       Reservation groupOf4 = new Reservation(4, 03, true);
       groupOf4.confirmReservation();
       groupOf4.informUser();

 }
}
```

```
Reservation confirmed
Please enjoy your meal!
Reservation denied
Unable to confirm reservation, please contact restaurant.
```