CONDITIONALS AND CONTROL FLOW

Introduction to Control Flow

Imagine we're writing a program that enrolls students in courses.

- *If* a student has completed the prerequisites, *then* they can enroll in a course.
- *Else*, they need to take the prerequisite courses.

They can't take Physics II without finishing Physics I.

We represent this kind of decision-making in our program using *conditional* or *control flow* statements. Before this point, our code runs line-by-line from the top down, but conditional statements allow us to be selective in which portions will run.

Conditional statements check a `boolean` condition and run a *block* of code depending on the condition. Curly braces mark the scope of a conditional block similar to a method or class.

Here's a complete conditional statement:

```
if (true) {
  System.out.println("Hello World!");
}
```

If the condition is `true`, then the block is run. So `Hello World!` is printed.

But suppose the condition is different:

```
if (false) {
  System.out.println("Hello World!");
}
```

If the condition is `false`, then the block does not run.

This code is also called *if-then* statements: "If `(condition)` is `true`, then do something".

If-Then Statement

The *if-then* statement is the most simple control flow we can write. It tests an expression for truth and executes some code based on it.

```
if (flip == 1) {

  System.out.println("Heads!");

}
```

- The `if` keyword marks the beginning of the conditional statement, followed by parentheses `()`.
- The parentheses hold a `boolean` datatype.

For the condition in parentheses we can also use variables that reference a boolean, or comparisons that evaluate to a boolean.

The boolean condition is followed by opening and closing curly braces that mark a block of code. This block runs if, and only if, the boolean is `true`.

```
boolean isValidPassword = true;

if (isValidPassword) {
  System.out.println("Password accepted!");
}
// Prints "Password accepted!"

int numberOfItemsInCart = 9;

if (numberOfItemsInCart > 12) {

  System.out.println("Express checkout not available");

}
// Nothing is printed.
```

If a conditional is brief we can omit the curly braces entirely:

```
if (true) System.out.println("Brevity is the soul of wit");
```

Note: Conditional statements do not end in a semicolon.

The code editor contains an `Order` class to track retail shipments.

Write an `if-then` statement that prints `High value item!` when `itemCost` is greater than `24.00`.

```
public class Order {

  public static void main(String[] args) {

    double itemCost = 30.99;

    // Write an if-then statement:
    if (itemCost > 24.00){
      System.out.println("High value item !");
    }
  }
}
```

```
High value item !
```

If-Then-Else

We've seen how to conditionally execute one block of code, but what if there are two possible blocks of code we'd like to execute?

Let's say *if* a student has the required prerequisite, *then* they enroll in the selected course, *else* they're enrolled in the prerequisite course instead.

We create an alternate conditional branch with the `else` keyword:

```
if (hasPrerequisite) {
  // Enroll in course
} else {
  // Enroll in prerequisite
}
```

This conditional statement ensures that exactly one code block will be run. If the condition, `hasPrerequisite`, is `false`, the block after `else` runs.

There are now two separate code blocks in our conditional statement. The first block runs if the condition evaluates to `true`, the second block runs if the condition evaluates to `false`.

This code is also called an *if-then-else* statement:

- If *condition* is true, then do something.
- Else, do a different thing.

Instructions

1.In the code editor, there is an `isFilled` value, that represents whether the order is ready to ship.

Write an if-then-else statement that:

- When `isFilled` is `true`, print `Shipping`.
- When `isFilled` is `false`, print `Order not ready`.

```
Public class Order {

 public static void main(String[] args) {

  boolean isFilled = false;

  // Write an if-then-else statement:

  if (isFilled)
  {
    System.out.println("ready for shipping.");
  }
  else {
        System.out.println("Order not ready .");
  }
 }
```

```
Order not ready .
```

If-Then-Else-If

The conditional structure we've learned can be chained together to check as many conditions as are required by our program. Imagine our program is now selecting the appropriate course for a student. We'll check their submission to find the correct course enrollment. The conditional statement now has multiple conditions that are evaluated from the top down:

```
String course = "Theatre";

if (course.equals("Biology")) {

  // Enroll in Biology course

} else if (course.equals("Algebra")) {

  // Enroll in Algebra course

} else if (course.equals("Theatre")) {

  // Enroll in Theatre course

} else {
  System.out.println("Course not found!");
}
```

The first condition to evaluate to `true` will have that code block run. Here's an example demonstrating the order:

```
int testScore = 72;

if (testScore >= 90) {

  System.out.println("A");

} else if (testScore >= 80) {

  System.out.println("B");
```

```
} else if (testScore >= 70) {

  System.out.println("C");

} else if (testScore >= 60) {

  System.out.println("D");

} else {

  System.out.println("F");

}
// prints: C
```

This chained conditional statement has two conditions that evaluate `true`. Because `testScore >= 70` comes before `testScore >= 60`, only the earlier code block is run.

Note: Only one of the code blocks will run.

Instructions

1. We need to calculate the shipping costs for our orders.

There's a new instance field, `String shipping`, that we use to calculate the cost.

Use a chained `if-then-else` to check for different values within the `calculateShipping()` method.

When the `shipping` instance field equals `"Regular"`, the method should return `0`.

When the `shipping` instance field equals `"Express"`, the method should return `1.75`.

Else the method should return `.50`.

```
public class Order {
  boolean isFilled;
  double billAmount;
  String shipping;

  public Order(boolean filled, double cost, String shippingMethod) {
    if (cost > 24.00) {
      System.out.println("High value item!");
    }
    isFilled = filled;
    billAmount = cost;
    shipping = shippingMethod;
  }

  public void ship() {
    if (isFilled) {
      System.out.println("Shipping");
      System.out.println("Shipping cost: " + calculateShipping());
    } else {
      System.out.println("Order not ready");
    }}

  public double calculateShipping() {
    // declare conditional statement here
    if (shipping.equals("Regular")){
          return 0;
    }else if(shipping.equals("Express")){
          return 1.75;
    } else {
          return 0.50;
    }}
     public static void main(String[] args) {
    // do not alter the main method!
    Order book = new Order(true, 9.99, "Express");
    Order chemistrySet = new Order(false, 72.50, "Regular");

    book.ship();
    chemistrySet.ship();
  }}
```

```
High value item!
Shipping
Shipping cost: 1.75
Order not ready
```

Nested Conditional Statements

We can create more complex conditional structures by creating *nested conditional statements*, which is created by placing conditional statements inside other conditional statements:

```
if (outer condition) {
   if (nested condition) {
     Instruction to execute if both conditions are true
   }
 }
```

When we implement nested conditional statements, the outer statement is evaluated first. If the outer condition is `true`, then the inner, nested statement is evaluated.

Let's create a program that helps us decide what to wear based on the weather:

```
int temp = 45;
boolean raining = true;

if (temp < 60) {
  System.out.println("Wear a jacket!");
  if (raining == true) {
    System.out.println("Bring your umbrella.");
  } else {
    System.out.println("Leave your umbrella home.");
  }
}
```

In the code snippet above, our compiler will check the condition in the first if-then statement: temp < 60. Since temp has a value of 45, this condition is true; therefore, our program will print Wear a jacket!.

Then, we'll evaluate the condition of the nested if-then statement: raining == true. This condition is also true, so Bring your umbrella is also printed to the screen.

Note that, if the first condition was false, the nested condition would not be evaluated.

Instructions

1. The company offers a temporary deal that, if the consumer uses the coupon "ship50", the company will reduce the express shipping price.

Let's rewrite the body of else-if statement from the last exercise. Inside the else-if statement, create a nested if-then statement that checks if couponCode equals "ship50".

If the nested condition is true, return the value .85.

If the condition is false, use a nested else statement to return the value 1.75.

```
public class Order {
  boolean isFilled;
  double billAmount;
  String shipping;
  String couponCode;

  public Order(boolean filled, double cost, String shippingMethod, String
coupon) {
    if (cost > 24.00) {
      System.out.println("High value item!");
    }
    isFilled = filled;
    billAmount = cost;
```

```
    shipping = shippingMethod;
    couponCode = coupon;
  }

  public void ship() {
    if (isFilled) {
      System.out.println("Shipping");
      System.out.println("Shipping cost: " + calculateShipping());
    } else {
      System.out.println("Order not ready");
    }
  }

  public double calculateShipping() {
    if (shipping.equals("Regular")) {
      return 0;
    } else if (shipping.equals("Express")) {
      // Add your code here
      if (couponCode.equals("ship50")) {
        return .85;
      } else {
        return 1.75;
      }
    } else {
      return .50;
    }
  }

  public static void main(String[] args) {
    // do not alter the main method!
    Order book = new Order(true, 9.99, "Express", "ship50");
    Order chemistrySet = new Order(false, 72.50, "Regular", "freeShipping");

    book.ship();
    chemistrySet.ship();
  }
}
```

```
High value item!
Shipping
Shipping cost: 0.85
Order not ready
```

Switch Statement

An alternative to chaining if-then-else conditions together is to use the `switch` statement. This conditional will check a given value against any number of conditions and run the code block where there is a match.

Here's an example of our course selection conditional as a `switch` statement instead:

```
String course = "History";

switch (course) {
  case "Algebra":
    // Enroll in Algebra
    break;
  case "Biology":
    // Enroll in Biology
    break;
  case "History":
    // Enroll in History
    break;
  case "Theatre":
    // Enroll in Theatre
    break;
  default:
    System.out.println("Course not found");
}
```

This example enrolls the student in History class by checking the value contained in the parentheses, `course`, against each of the `case` labels. If the value after the case label matches the value within the parentheses, the *switch block* is run.

In the above example, `course` references the string `"History"`, which matches `case "History":`.

When no value matches, the `default` block runs. Think of this as the `else` equivalent.

Switch blocks are different than other code blocks because they are not marked by curly braces and we use the `break` keyword to exit the switch statement.

Without `break`, code below the matching `case` label is run, *including code under other case labels*, which is rarely the desired behavior.

```
String course = "Biology";

switch (course) {
  case "Algebra":
    // Enroll in Algebra
  case "Biology":
    // Enroll in Biology
  case "History":
    // Enroll in History
  case "Theatre":
    // Enroll in Theatre
  default:
    System.out.println("Course not found");
}
// enrolls student in Biology... AND History and Theatre!
```

Instructions

1. We'll rewrite the `calculateShipping()` method so it uses a `switch` statement instead.

There's an uninitialized variable `shippingCost` in `calculateShipping()`. Assign the correct value to `shippingCost` using a `switch` statement:

We'll check the value of the instance field `shipping`.

- When `shipping` matches `"Regular"`, `shippingCost` should be `0`.
- When `shipping` matches `"Express"`, `shippingCost` should be `1.75`.
- The default should assign `.50` to `shippingCost`.

Make sure the method returns `shippingCost` after the `switch` statement.

```java
public class Order {
  boolean isFilled;
  double billAmount;
  String shipping;

  public Order(boolean filled, double cost, String shippingMethod) {
    if (cost > 24.00) {
      System.out.println("High value item!");
    }
    isFilled = filled;
    billAmount = cost;
    shipping = shippingMethod;
  }

  public void ship() {
    if (isFilled) {
      System.out.println("Shipping");
      System.out.println("Shipping cost: " + calculateShipping());
    } else {
      System.out.println("Order not ready");
    }
  }

  public double calculateShipping() {
    double shippingCost;
    // declare switch statement here
    switch (shipping) {
      case "Regular": shippingCost = 0; break;
      case "Express": shippingCost = 1.75; break;
      default:
        shippingCost = 0.50;
    }
    return shippingCost;
  }
  public static void main(String[] args) {
    // do not alter the main method!
    Order book = new Order(true, 9.99, "Express");
    Order chemistrySet = new Order(false, 72.50, "Regular");

    book.ship();
    chemistrySet.ship();
  }
}
```

```
High value item!
Shipping
Shipping cost: 1.75
Order not ready
```

Review

Before this lesson, our code executed from top to bottom, line by line.

Conditional statements add branching paths to our programs. We use Conditionals to make decisions in the program so that different inputs will produce different results.

Conditionals have the general structure:

```
if (condition) {
   // consequent path
} else {
   // alternative path
}
```

Specific conditional statements have the following behavior:

if-then:code block runs if condition is true

if-then-else:one block runs if conditions is true----another block runs if condition is false

if-then-else chained:same as if-then but an arbitrary number of conditions

switch: switch block runs if condition value matches case value

Instructions

Our complete Order program is in the text editor but the main() method is empty.

Create different Order instances and see if you can run the code in all the different conditional blocks!

```
public class Order {
  boolean isFilled;
  double billAmount;
  String shipping;

  public Order(boolean filled, double cost, String shippingMethod) {
```

```
  if (cost > 24.00) {
    System.out.println("High value item!");
  } else {
    System.out.println("Low value item!");
  }
  isFilled = filled;
  billAmount = cost;
  shipping = shippingMethod;
}

public void ship() {
  if (isFilled) {
    System.out.println("Shipping");
  } else {
    System.out.println("Order not ready");
  }

  double shippingCost = calculateShipping();

  System.out.println("Shipping cost: ");
  System.out.println(shippingCost);
}

public double calculateShipping() {
  double shippingCost;
  switch (shipping) {
    case "Regular":
      shippingCost = 0;
      break;
    case "Express":
      shippingCost = 1.75;
      break;
    default:
      shippingCost = .50;
  }
  return shippingCost;
}

public static void main(String[] args) {
  // create instances and call methods here!
  Order o1 = new Order(false, 35.00, "Express");
  o1.calculateShipping();
  o1.ship();
}}
```

```
High value item!
Order not ready
Shipping cost:
1.75
```