Conditional Statements and Program Flow Control

Mitsu Ogihara

Department of Computer Science
University of Miami

Table of Contents

Outline

Conditions

- A condition is a statement, a variable, or a constant that evaluates to a boolean value (true or false)
- The name "Boolean" is taken after George Boole, a 19th century mathematician who did fundamental work in logic and algebra
- A boolean is a primitive data type (as opposed to an object data type), has only two possible values
  - `true`
  - `false`
- The two values are opposite to each other
- To represent a boolean value you need just one bit

Declaration and Printing of Boolean

- To declare a boolean variable: `boolean VARIABLE-NAME;`
- A boolean variable can be assigned a value using an assignment (that is, '=')
- Java uses the string literals `"true"` and `"false"` respectively for printing the boolean values `true` and `false`

```java
public class BooleanPrint {
  public static void main( String[] args ) {
    boolean t = true;
    boolean f = false;
    System.out.println( t );
    System.out.println( f );
  }
}
```

Assigning values

Declaration and Printing of Boolean

- To declare a boolean variable: `boolean VARIABLE-NAME;`
- A boolean variable can be assigned a value using an assignment (that is, '=')
- Java uses the string literals `"true"` and `"false"` respectively for printing the boolean values `true` and `false`

```java
public class BooleanPrint {
  public static void main( String[] args ) {
    boolean t = true;
    boolean f = false;
    System.out.println( t );
    System.out.println( f );
  }
}
```

Printing the values

Declaration and Printing of Boolean

- To declare a boolean variable: `boolean VARIABLE-NAME;`
- A boolean variable can be assigned a value using an assignment (that is, '=')
- Java uses the string literals `"true"` and `"false"` respectively for printing the boolean values `true` and `false`

```java
public class BooleanPrint {
  public static void main( String[] args ) {
    boolean t = true;
    boolean f = false;
    System.out.println( t );
    System.out.println( f );
  }
}
```

The output is:

```
true
false
```

Arithmetic on Boolean Type

There are three operations on boolean types.
Let `x` be a boolean type.

- The negation : flip between `true` and `false`
  - To use attach `!` before the boolean type
  - `!x` is equal to `false` if `x` is equal to `true`
  - `!x` is equal to `true` if `x` is equal to `false`
- The logical-or : "either □ or □" of boolean types
  - Place two vertical signs `||` between the two boolean types
  - More than two boolean types can be connected
- The logical-and : "both □ and □" of boolean types
  - Place `&&` between the two boolean types
  - More than two boolean types can be connected

Boolean Connectives

Let's take a look at the use of the three boolean operations

```
public class BooleanConnectives {
  public static void main( String[] args ) {
    ////////////////////////////////////////////////////
    // NOT
    ////////////////////////////////////////////////////
    System.out.println( "---------- NOT --------------" );
    System.out.println( "!true is " + (!true) );
    System.out.println( "!false is " + (!false) );
    System.out.println( "!!true is " + (!(!true)) );
    System.out.println( "!!false is " + (!(!false)) );
```

For all the combinations, the `String` part and the value part are connected

Boolean Connectives (cont'd)

```
    //////////////////////////////////////////////////
    // AND
    //////////////////////////////////////////////////
    System.out.println( "---------- AND --------------" );
    System.out.print( "true && true is " );
    System.out.println( true && true );
    System.out.print( "true && false is " );
    System.out.println( true && false );
    System.out.print( "false && true is " );
    System.out.println( false && true );
    System.out.print( "false && false is " );
    System.out.println( false && false );
```

Four different combinations
These are for printing the `String` objects

Boolean Connectives (cont'd)

```
    ////////////////////////////////////////////////////
    // AND
    ////////////////////////////////////////////////////
    System.out.println( "---------- AND --------------" );
    System.out.print( "true && true is " );
    System.out.println( true && true );
    System.out.print( "true && false is " );
    System.out.println( true && false );
    System.out.print( "false && true is " );
    System.out.println( false && true );
    System.out.print( "false && false is " );
    System.out.println( false && false );
```

The values

Boolean Connectives (cont'd)

```
    ////////////////////////////////////////////////////
    // OR
    ////////////////////////////////////////////////////
    System.out.println( "---------- OR ---------------" );
    System.out.print( "true || true is " );
    System.out.println( true || true );
    System.out.print( "true || false is " );
    System.out.println( true || false );
    System.out.print( "false || true is " );
    System.out.println( false || true );
    System.out.print( "false || false is " );
    System.out.println( false || false );
  }
}
```

Four different combinations
These are for printing the `String` objects

Boolean Connectives (cont'd)

```
    ///////////////////////////////////////////////////
    // OR
    ///////////////////////////////////////////////////
    System.out.println( "---------- OR ---------------" );
    System.out.print( "true || true is " );
    System.out.println( true || true );
    System.out.print( "true || false is " );
    System.out.println( true || false );
    System.out.print( "false || true is " );
    System.out.println( false || true );
    System.out.print( "false || false is " );
    System.out.println( false || false );
  }
}
```

The values

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Like the other type-specific next methods, InputMismatchException may occur if unexpected text is typed when a boolean is expected

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Like the other type-specific next methods, InputMismatchException may occur if unexpected text is typed when a boolean is expected

```
import java.util.Scanner;
// receive two boolean values and compute their AND and OR
public class ReadBoolean {
  public static void main( String[] args ) {
    Scanner console;
    boolean value1, value2, conj, disj;
    console = new Scanner( System.in );
    System.out.print( "enter boolean value 1: " );
    value1 = console.nextBoolean();
    System.out.print( "enter boolean value 2: " );
    value2 = console.nextBoolean();
    conj = value1 && value2;
    disj = value1 || value2;
    System.out.println( "The values are " + value1 + " and " + value2 );
    System.out.println( "Their conjunction (AND) is " + conj );
    System.out.println( "Their disjunction (OR) is " + disj );
  }
}
```

Declaring variables

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Like the other type-specific next methods, InputMismatchException may occur if unexpected text is typed when a boolean is expected

```java
import java.util.Scanner;
// receive two boolean values and compute their AND and OR
public class ReadBoolean {
  public static void main( String[] args ) {
    Scanner console;
    boolean value1, value2, conj, disj;
    console = new Scanner( System.in );
    System.out.print( "enter boolean value 1: " );
    value1 = console.nextBoolean();
    System.out.print( "enter boolean value 2: " );
    value2 = console.nextBoolean();
    conj = value1 && value2;
    disj = value1 || value2;
    System.out.println( "The values are " + value1 + " and " + value2 );
    System.out.println( "Their conjunction (AND) is " + conj );
    System.out.println( "Their disjunction (OR) is " + disj );
  }
}
```

Scanner creation and import

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Like the other type-specific next methods, InputMismatchException may occur if unexpected text is typed when a boolean is expected

```java
import java.util.Scanner;
// receive two boolean values and compute their AND and OR
public class ReadBoolean {
  public static void main( String[] args ) {
    Scanner console;
    boolean value1, value2, conj, disj;
    console = new Scanner( System.in );
    System.out.print( "enter boolean value 1: " );
    value1 = console.nextBoolean();
    System.out.print( "enter boolean value 2: " );
    value2 = console.nextBoolean();
    conj = value1 && value2;
    disj = value1 || value2;
    System.out.println( "The values are " + value1 + " and " + value2 );
    System.out.println( "Their conjunction (AND) is " + conj );
    System.out.println( "Their disjunction (OR) is " + disj );
  }
}
```

Reading boolean values using the Scanner

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Like the other type-specific next methods, InputMismatchException may occur if unexpected text is typed when a boolean is expected

```
import java.util.Scanner;
// receive two boolean values and compute their AND and OR
public class ReadBoolean {
  public static void main( String[] args ) {
    Scanner console;
    boolean value1, value2, conj, disj;
    console = new Scanner( System.in );
    System.out.print( "enter boolean value 1: " );
    value1 = console.nextBoolean();
    System.out.print( "enter boolean value 2: " );
    value2 = console.nextBoolean();
    conj = value1 && value2;
    disj = value1 || value2;
    System.out.println( "The values are " + value1 + " and " + value2 );
    System.out.println( "Their conjunction (AND) is " + conj );
    System.out.println( "Their disjunction (OR) is " + disj );
  }
}
```

Computing the AND and the OR of the two values

Read a Boolean value via a Scanner

We can use `nextBoolean()` to read a boolean via a Scanner

Like the other type-specific next methods, InputMismatchException may occur if unexpected text is typed when a boolean is expected

```java
import java.util.Scanner;
// receive two boolean values and compute their AND and OR
public class ReadBoolean {
  public static void main( String[] args ) {
    Scanner console;
    boolean value1, value2, conj, disj;
    console = new Scanner( System.in );
    System.out.print( "enter boolean value 1: " );
    value1 = console.nextBoolean();
    System.out.print( "enter boolean value 2: " );
    value2 = console.nextBoolean();
    conj = value1 && value2;
    disj = value1 || value2;
    System.out.println( "The values are " + value1 + " and " + value2 );
    System.out.println( "Their conjunction (AND) is " + conj );
    System.out.println( "Their disjunction (OR) is " + disj );
  }
}
```

Presenting the results

Two Things to Remember About Boolean Formula Evaluation

1. **Combining ANDs and ORs**
   When more than one type of operators from `||`, `&&`, and `!` appears in a boolean formula, they are treated similar to $+$, $*$, and $-$ (the negative sign) in the number arithmetic
2. **Short-circuited Evaluation** Boolean formulas are evaluated from left to right. Evaluation terminates as soon as the value has been determined without evaluation the rest of the formula

Outline

Generating Booleans

Booleans can be generated by combining the following rules:

- The boolean literals `true` and `false` are conditions
- The equality and inequality tests, `==` and `!=`, are conditions
- The comparisons of numbers, `<`, `<=`, `>=`, `>`, are conditions
- *Call to methods that return a boolean are conditions, e.g., equals, startsWith, and endsWith of Class String*
- Conditions with `!` attached in front are conditions
- Conditions connected with `||` and/or `&&` are conditions

Equality Tests and Comparisons

- Equality and inequality tests `==` and `!=`
  ··· work for primitive data types (not for object types, including String)
  ··· work when testing null (i.e., "undefined") by way of `== null` and `!= null`
- Comparisons `<, >, <=, >=`
  ··· work for primitive number types

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```java
import java.util. * ;
// various comparisons
public class Comparisons0 {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    //------------- int
    System.out.print( "Enter two int: " );
    int number1 = console.nextInt();
    int number2 = console.nextInt();
    System.out.println( "number1 = " + number1 + ", number2 = " + number2 );
    System.out.println( "number1 == number2 is " + ( number1 == number2 ) );
    System.out.println( "number1 != number2 is " + ( number1 != number2 ) );
    System.out.println( "number1 >= number2 is " + ( number1 >= number2 ) );
    System.out.println( "number1 > number2 is " + ( number1 > number2 ) );
    System.out.println( "number1 <= number2 is " + ( number1 <= number2 ) );
    System.out.println( "number1 < number2 is " + ( number1 < number2 ) );
```

Scanner creation

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```
import java.util. * ;
// various comparisons
public class Comparisons0 {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    //------------- int
    System.out.print( "Enter two int: " );
    int number1 = console.nextInt();
    int number2 = console.nextInt();
    System.out.println( "number1 = " + number1 + ", number2 = " + number2 );
    System.out.println( "number1 == number2 is " + ( number1 == number2 ) );
    System.out.println( "number1 != number2 is " + ( number1 != number2 ) );
    System.out.println( "number1 >= number2 is " + ( number1 >= number2 ) );
    System.out.println( "number1 > number2 is " + ( number1 > number2 ) );
    System.out.println( "number1 <= number2 is " + ( number1 <= number2 ) );
    System.out.println( "number1 < number2 is " + ( number1 < number2 ) );
```

Receive two integers

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```java
import java.util. * ;
// various comparisons
public class Comparisons0 {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    //------------- int
    System.out.print( "Enter two int: " );
    int number1 = console.nextInt();
    int number2 = console.nextInt();
    System.out.println( "number1 = " + number1 + ", number2 = " + number2 );
    System.out.println( "number1 == number2 is " + ( number1 == number2 ) );
    System.out.println( "number1 != number2 is " + ( number1 != number2 ) );
    System.out.println( "number1 >= number2 is " + ( number1 >= number2 ) );
    System.out.println( "number1 > number2 is " + ( number1 > number2 ) );
    System.out.println( "number1 <= number2 is " + ( number1 <= number2 ) );
    System.out.println( "number1 < number2 is " + ( number1 < number2 ) );
```

Execute comparisons

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```
    //------------- double
    System.out.print( "Enter two double: " );
    double real1 = console.nextDouble();
    double real2 = console.nextDouble();
    System.out.println( "real1 = " + real1 + ", real2 = " + real2 );
    System.out.println( "real1 == real2 is " + ( real1 == real2 ) );
    System.out.println( "real1 != real2 is " + ( real1 != real2 ) );
    System.out.println( "real1 >= real2 is " + ( real1 >= real2 ) );
    System.out.println( "real1 > real2 is " + ( real1 > real2 ) );
    System.out.println( "real1 <= real2 is " + ( real1 <= real2 ) );
    System.out.println( "real1 < real2 is " + ( real1 < real2 ) );
    //------------- String
    System.out.print( "Enter two Strings: " );
    String word1 = console.next();
    String word2 = console.next();
    System.out.println( "word1 = " + word1 + ", word2 = " + word2 );
    System.out.println( "word1 == word2 is " + ( word1 == word2 ) );
    System.out.println( "word1 != word2 is " + ( word1 != word2 ) );
  }
}
```

Receive two real numbers

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```
    //------------- double
    System.out.print( "Enter two double: " );
    double real1 = console.nextDouble();
    double real2 = console.nextDouble();
    System.out.println( "real1 = " + real1 + ", real2 = " + real2 );
    System.out.println( "real1 == real2 is " + ( real1 == real2 ) );
    System.out.println( "real1 != real2 is " + ( real1 != real2 ) );
    System.out.println( "real1 >= real2 is " + ( real1 >= real2 ) );
    System.out.println( "real1 > real2 is " + ( real1 > real2 ) );
    System.out.println( "real1 <= real2 is " + ( real1 <= real2 ) );
    System.out.println( "real1 < real2 is " + ( real1 < real2 ) );
    //------------- String
    System.out.print( "Enter two Strings: " );
    String word1 = console.next();
    String word2 = console.next();
    System.out.println( "word1 = " + word1 + ", word2 = " + word2 );
    System.out.println( "word1 == word2 is " + ( word1 == word2 ) );
    System.out.println( "word1 != word2 is " + ( word1 != word2 ) );
  }
}
```

Execute comparisons

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```java
    //------------- double
    System.out.print( "Enter two double: " );
    double real1 = console.nextDouble();
    double real2 = console.nextDouble();
    System.out.println( "real1 = " + real1 + ", real2 = " + real2 );
    System.out.println( "real1 == real2 is " + ( real1 == real2 ) );
    System.out.println( "real1 != real2 is " + ( real1 != real2 ) );
    System.out.println( "real1 >= real2 is " + ( real1 >= real2 ) );
    System.out.println( "real1 > real2 is " + ( real1 > real2 ) );
    System.out.println( "real1 <= real2 is " + ( real1 <= real2 ) );
    System.out.println( "real1 < real2 is " + ( real1 < real2 ) );
    //------------- String
    System.out.print( "Enter two Strings: " );
    String word1 = console.next();
    String word2 = console.next();
    System.out.println( "word1 = " + word1 + ", word2 = " + word2 );
    System.out.println( "word1 == word2 is " + ( word1 == word2 ) );
    System.out.println( "word1 != word2 is " + ( word1 != word2 ) );
  }
}
```

Receive two Strings

Condition Generation Examples

Receive a pair of int values, a pair of double values, and a pair of Strings, and execute comparison

```java
    //------------- double
    System.out.print( "Enter two double: " );
    double real1 = console.nextDouble();
    double real2 = console.nextDouble();
    System.out.println( "real1 = " + real1 + ", real2 = " + real2 );
    System.out.println( "real1 == real2 is " + ( real1 == real2 ) );
    System.out.println( "real1 != real2 is " + ( real1 != real2 ) );
    System.out.println( "real1 >= real2 is " + ( real1 >= real2 ) );
    System.out.println( "real1 > real2 is " + ( real1 > real2 ) );
    System.out.println( "real1 <= real2 is " + ( real1 <= real2 ) );
    System.out.println( "real1 < real2 is " + ( real1 < real2 ) );
    //------------- String
    System.out.print( "Enter two Strings: " );
    String word1 = console.next();
    String word2 = console.next();
    System.out.println( "word1 = " + word1 + ", word2 = " + word2 );
    System.out.println( "word1 == word2 is " + ( word1 == word2 ) );
    System.out.println( "word1 != word2 is " + ( word1 != word2 ) );
  }
}
```

Execute comparisons

Table of Contents

Outline

The Use of `if`

The structure of an if statement is

```
if ( CONDITION ) {
  some statements
}
```

Example 1: Temperature01.java

```java
import java.util.Scanner;
// ask about temperature and respond
public class Temperature01 {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    //-- prompt answer
    System.out.print( "What is the average high temperature in "
      + "August in your area? : " );
    double temp = console.nextDouble();
    //-- response
    if ( temp > 90.0 ) {
      System.out.println( "Wow! That must be very hot!" );
    }
  }
}
```

The print statement is executed if and only if the temperature entered is strictly greater than 90.0

Example 2: Temperature02.java

```java
import java.util.Scanner;
// ask about temperature and respond
public class Temperature02 {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    //-- prompt answer
    System.out.print( "What is the average high temperature in "
        + "August in your area? : " );
    double temp = console.nextDouble();
    //-- response no.1
    if ( temp > 90.0 ) {
      System.out.println( "Wow! That must be very hot!" );
    }
    //-- response no.2
    if ( temp <= 70.0 ) {
      System.out.println( "Wow! That must be very cold!" );
    }
  }
}
```

This print statement is executed if and only if the temperature entered is strictly greater than 90.0

Example 2: Temperature02.java

```java
import java.util.Scanner;
// ask about temperature and respond
public class Temperature02 {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    //-- prompt answer
    System.out.print( "What is the average high temperature in "
        + "August in your area? : " );
    double temp = console.nextDouble();
    //-- response no.1
    if ( temp > 90.0 ) {
      System.out.println( "Wow! That must be very hot!" );
    }
    //-- response no.2
    if ( temp <= 70.0 ) {
      System.out.println( "Wow! That must be very cold!" );
    }
  }
}
```

This print statement is executed if and only if the temperature entered is less than or equal to 70.0

Example 3: Temperature03.java

```java
    double humidity = console.nextDouble();
    //-- response no.1
    if ( temp >= 90.0 && humidity >= 90.0 ) {
      System.out.println( "Wow! That must be hot and humid!" );
    }
    //-- response no.2
    if ( temp >= 90.0 && humidity <= 50.0 ) {
      System.out.println( "Wow! That must be hot and dry!" );
    }
    //-- response no.3
    if ( temp <= 70.0 ) {
      System.out.println( "Wow! That must be cool!" );
    }
    //-- response no.4
    if ( temp > 70.0 && humidity < 90.0 ) {
      System.out.println( "Wow! That must be very comfortable!" );
    }
  }
}
```

Print only if temperature is greater than or equal to 90.0 and humidity is greater than or equal to 90.0

Example 3: Temperature03.java

```java
    double humidity = console.nextDouble();
    //-- response no.1
    if ( temp >= 90.0 && humidity >= 90.0 ) {
      System.out.println( "Wow! That must be hot and humid!" );
    }
    //-- response no.2
    if ( temp >= 90.0 && humidity <= 50.0 ) {
      System.out.println( "Wow! That must be hot and dry!" );
    }
    //-- response no.3
    if ( temp <= 70.0 ) {
      System.out.println( "Wow! That must be cool!" );
    }
    //-- response no.4
    if ( temp > 70.0 && humidity < 90.0 ) {
      System.out.println( "Wow! That must be very comfortable!" );
    }
  }
}
```

Print only if temperature is greater than or equal to 90.0 and humidity is less than or equal to 50.0

Example 3: Temperature03.java

```
    double humidity = console.nextDouble();
    //-- response no.1
    if ( temp >= 90.0 && humidity >= 90.0 ) {
      System.out.println( "Wow! That must be hot and humid!" );
    }
    //-- response no.2
    if ( temp >= 90.0 && humidity <= 50.0 ) {
      System.out.println( "Wow! That must be hot and dry!" );
    }
    //-- response no.3
    if ( temp <= 70.0 ) {
      System.out.println( "Wow! That must be cool!" );
    }
    //-- response no.4
    if ( temp > 70.0 && humidity < 90.0 ) {
      System.out.println( "Wow! That must be very comfortable!" );
    }
  }
}
```

Print only if temperature is less than or equal to 70.0

Example 3: Temperature03.java

```java
      double humidity = console.nextDouble();
      //-- response no.1
      if ( temp >= 90.0 && humidity >= 90.0 ) {
        System.out.println( "Wow! That must be hot and humid!" );
      }
      //-- response no.2
      if ( temp >= 90.0 && humidity <= 50.0 ) {
        System.out.println( "Wow! That must be hot and dry!" );
      }
      //-- response no.3
      if ( temp <= 70.0 ) {
        System.out.println( "Wow! That must be cool!" );
      }
      //-- response no.4
      if ( temp > 70.0 && humidity < 90.0 ) {
        System.out.println( "Wow! That must be very comfortable!" );
      }
    }
}
```

Print only if temperature is greater than 70.0 and the humanity is less than 90.0

Example 3: Temperature03.java

```
    double humidity = console.nextDouble();
    //-- response no.1
    if ( temp >= 90.0 && humidity >= 90.0 ) {
      System.out.println( "Wow! That must be hot and humid!" );
    }
    //-- response no.2
    if ( temp >= 90.0 && humidity <= 50.0 ) {
      System.out.println( "Wow! That must be hot and dry!" );
    }
    //-- response no.3
    if ( temp <= 70.0 ) {
      System.out.println( "Wow! That must be cool!" );
    }
    //-- response no.4
    if ( temp > 70.0 && humidity < 90.0 ) {
      System.out.println( "Wow! That must be very comfortable!" );
    }
  }
}
```

No print statement will be executed if

```
temp > 70.0 && temp < 90.0 && humidity >= 90.0
```

Example 4: Color Selection

Ask user to select one of four colors and provide response

```java
import java.util.Scanner;
// ask about a color and respond
public class ColorSelection {
  public static void main( String[] args ) {
    //-- scanner
    Scanner console = new Scanner( System.in );
    System.out.println( "What is your favorite color?" );
    System.out.println( "1. Orange, 2. Green, 3. Yellow, 4. Blue" );
    System.out.print( "Select from 1 to 4 : " );
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    if ( answer >= 1 && answer <= 4 ) {
      System.out.println( "Your choice " + answer + " is great." );
    }
    if ( answer >= 1 && answer <= 2 ) {
      System.out.println( "It is a UM color!" );
    }
  }
}
```

Print when an invalid choice is made

Example 4: Color Selection

Ask user to select one of four colors and provide response

```java
import java.util.Scanner;
// ask about a color and respond
public class ColorSelection {
  public static void main( String[] args ) {
    //-- scanner
    Scanner console = new Scanner( System.in );
    System.out.println( "What is your favorite color?" );
    System.out.println( "1. Orange, 2. Green, 3. Yellow, 4. Blue" );
    System.out.print( "Select from 1 to 4 : " );
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    if ( answer >= 1 && answer <= 4 ) {
      System.out.println( "Your choice " + answer + " is great." );
    }
    if ( answer >= 1 && answer <= 2 ) {
      System.out.println( "It is a UM color!" );
    }
  }
}
```

Print when a valid choice is made

Example 4: Color Selection

Ask user to select one of four colors and provide response

```java
import java.util.Scanner;
// ask about a color and respond
public class ColorSelection {
  public static void main( String[] args ) {
    //-- scanner
    Scanner console = new Scanner( System.in );
    System.out.println( "What is your favorite color?" );
    System.out.println( "1. Orange, 2. Green, 3. Yellow, 4. Blue" );
    System.out.print( "Select from 1 to 4 : " );
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    if ( answer >= 1 && answer <= 4 ) {
      System.out.println( "Your choice " + answer + " is great." );
    }
    if ( answer >= 1 && answer <= 2 ) {
      System.out.println( "It is a UM color!" );
    }
  }
}
```

Print when a UM color is chosen

Outline

If-else Statements

In Java (and in many other programming languages), `else` can be used to mean "otherwise,"

Its syntax is:

```
if (CONDITIONAL-1) { STATEMENTS-1 }
else if (CONDITIONAL-2) { STATEMENTS-2 }
...
else if (CONDITIONAL-k) { STATEMENTS-k }
else { STATEMENTS-(k+1) }
```

If-else Statements

In Java (and in many other programming languages), `else` can be used to mean "otherwise,"

Its syntax is:

```
if (CONDITIONAL-1) { STATEMENTS-1 }
else if (CONDITIONAL-2) { STATEMENTS-2 }
...
else if (CONDITIONAL-k) { STATEMENTS-k }
else { STATEMENTS-(k+1) }
```

Here `k` be any positive integer (possibly 1) and the last `else` line is optional.

If-else Statements

In Java (and in many other programming languages), `else` can be used to mean "otherwise,"

Its syntax is:

```
if (CONDITIONAL-1) { STATEMENTS-1 }
else if (CONDITIONAL-2) { STATEMENTS-2 }
...
else if (CONDITIONAL-k) { STATEMENTS-k }
else { STATEMENTS-(k+1) }
```

Here `k` be any positive integer (possibly 1) and the last `else` line is optional. When run, `m` such that `CONDITIONAL-m` is `true` is found and `STATEMENTS-m` are executed; `STATEMENTS-(k+1)` are executed if no such `m` exists and if that line indeed exists

Color Selection Revisited

Ask user to select one of four colors and provide response

```
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else if ( answer >= 3 && answer <= 4 ) {
      System.out.println( "Your choice " + answer + " is great, but " );
      System.out.println( "it is not a UM color!" );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
    }
  }
}
```

Print when an invalid choice is made

Color Selection Revisited

Ask user to select one of four colors and provide response

```
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else if ( answer >= 3 && answer <= 4 ) {
      System.out.println( "Your choice " + answer + " is great, but " );
      System.out.println( "it is not a UM color!" );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
    }
  }
}
```

Print when a valid choice is made and the color is a UM color

Color Selection Revisited

Ask user to select one of four colors and provide response

```
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else if ( answer >= 3 && answer <= 4 ) {
      System.out.println( "Your choice " + answer + " is great, but " );
      System.out.println( "it is not a UM color!" );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
    }
  }
}
```

Print when a valid choice is made and the color is NOT a UM color

The Utility of else

`else` allows to simplify the conditionals.

Without using `else`, `if (CONDITIONAL-1) { STATEMENTS-1 }`

```
else if (CONDITIONAL-2) { STATEMENTS-2 }
...
else if (CONDITIONAL-k) { STATEMENTS-k }
else { STATEMENTS-(k+1) }
```

will look like:

The Utility of else

`else` allows to simplify the conditionals.

Without using `else`,
```
if (CONDITIONAL-1) { STATEMENTS-1 }
else if (CONDITIONAL-2) { STATEMENTS-2 }
...
else if (CONDITIONAL-k) { STATEMENTS-k }
else { STATEMENTS-(k+1) }
```
will look like:

```
if (CONDITIONAL-1) { STATEMENTS-1 }
if (!CONDITIONAL-1 && CONDITIONAL-2) { STATEMENTS-2 }
...
if (!CONDITIONAL-1 && !CONDITIONAL-2 ...  &&
    !CONDITIONAL-(k-1) && CONDITIONAL-k) { STATEMENTS-k
}
if (!CONDITIONAL-1 && !CONDITIONAL-2 ...  &&
    !CONDITIONAL-k && CONDITIONAL-(k+1))
        { STATEMENTS-(k+1) }
```

Outline

Nesting of if-else Statements

By "nesting" we mean that there is an if or if-else statement in the block under if, else-if, or else

Color Selection Revisited: ColorSelectionNested.java

Ask user to select one of four colors and provide response

```java
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
      if ( answer == 1 || answer == 2 ) {
        System.out.println( "It is a UM color!" );
        if ( answer == 1 ) {
          System.out.println( "It is my favorite color!" );
        }
      }
      else {
        System.out.println( "Good choice." );
      }
    }
  }
}
```

Print when an invalid choice is made

Color Selection Revisited: ColorSelectionNested.java

Ask user to select one of four colors and provide response

```java
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
      if ( answer == 1 || answer == 2 ) {
        System.out.println( "It is a UM color!" );
        if ( answer == 1 ) {
          System.out.println( "It is my favorite color!" );
        }
      }
      else {
        System.out.println( "Good choice." );
      }
    }
  }
}
```

This "else" captures all the valid cases

Color Selection Revisited: ColorSelectionNested.java

Ask user to select one of four colors and provide response

```
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
      if ( answer == 1 || answer == 2 ) {
        System.out.println( "It is a UM color!" );
        if ( answer == 1 ) {
          System.out.println( "It is my favorite color!" );
        }
      }
      else {
        System.out.println( "Good choice." );
      }
    }
  }
}
```

Print when a valid choice is made and the color is NOT a UM color

Color Selection Revisited: ColorSelectionNested.java

Ask user to select one of four colors and provide response

```
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
      if ( answer == 1 || answer == 2 ) {
        System.out.println( "It is a UM color!" );
        if ( answer == 1 ) {
          System.out.println( "It is my favorite color!" );
        }
      }
      else {
        System.out.println( "Good choice." );
      }
    }
  }
}
```

Valid, but not UM color

Color Selection Revisited: ColorSelectionNested.java

Ask user to select one of four colors and provide response

```
    int answer = console.nextInt();
    if ( answer < 1 || answer > 4 ) {
      System.out.println( "!!!Your choice " + answer + " is invalid." );
    }
    else {
      System.out.println( "Your choice " + answer + " is great." );
      if ( answer == 1 || answer == 2 ) {
        System.out.println( "It is a UM color!" );
        if ( answer == 1 ) {
          System.out.println( "It is my favorite color!" );
        }
      }
      else {
        System.out.println( "Good choice." );
      }
    }
  }
}
```

Orange

Table of Contents

A Cumulative Algorithm

A cumulative algorithm is a strategy for obtaining the result by processing a series of data elements one after the other with the following general principle:

- Initialize a set of variables, say *S*, some of which are to hold the values we wish to compute ultimately
- Start receiving the elements one after the other
- If there is a new data element
    - update the variables *S* with some "small" amount of computation
- Otherwise, the computation is complete.

An Illustrative Example: Computing the Running Total

- The program prompts the user to enter the number of data
- Then runs a for-loop to receive the data
- The program outputs the total

Code

```java
import java.util.Scanner;
public class CumulativeSimple {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    double total = 0, input;
    System.out.print( "Enter the number of data: " );
    int nData = console.nextInt();
    for ( int i = 1; i <= nData; i ++ ) {
      System.out.print( "Enter the data #" + i + ": " );
      input = console.nextDouble();
      total += input;
    }
    System.out.println( "The total is " + total );
  }
}
```

Initialize the sum as 0; `input` is the variable to store the input from the user

Code

```java
import java.util.Scanner;
public class CumulativeSimple {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    double total = 0, input;
    System.out.print( "Enter the number of data: " );
    int nData = console.nextInt();
    for ( int i = 1; i <= nData; i ++ ) {
      System.out.print( "Enter the data #" + i + ": " );
      input = console.nextDouble();
      total += input;
    }
    System.out.println( "The total is " + total );
  }
}
```

Receive the number of data to process

Code

```java
import java.util.Scanner;
public class CumulativeSimple {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    double total = 0, input;
    System.out.print( "Enter the number of data: " );
    int nData = console.nextInt();
    for ( int i = 1; i <= nData; i ++ ) {
      System.out.print( "Enter the data #" + i + ": " );
      input = console.nextDouble();
      total += input;
    }
    System.out.println( "The total is " + total );
  }
}
```

Run a for-loop

Code

```java
import java.util.Scanner;
public class CumulativeSimple {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    double total = 0, input;
    System.out.print( "Enter the number of data: " );
    int nData = console.nextInt();
    for ( int i = 1; i <= nData; i ++ ) {
      System.out.print( "Enter the data #" + i + ": " );
      input = console.nextDouble();
      total += input;
    }
    System.out.println( "The total is " + total );
  }
}
```

Prompt the user, receive data, and update the total

Code

```java
import java.util.Scanner;
public class CumulativeSimple {
  public static void main( String[] args ) {
    Scanner console = new Scanner( System.in );
    double total = 0, input;
    System.out.print( "Enter the number of data: " );
    int nData = console.nextInt();
    for ( int i = 1; i <= nData; i ++ ) {
      System.out.print( "Enter the data #" + i + ": " );
      input = console.nextDouble();
      total += input;
    }
    System.out.println( "The total is " + total );
  }
}
```

Print the total

Computing the Total, Max, Min, and Average

In addition to compute the total, we must compute the maximum, the minimum, and the average of all the numbers entered

Computing the Total, Max, Min, and Average

In addition to compute the total, we must compute the maximum, the minimum, and the average of all the numbers entered

In general, the maximum can be updated by:

```
if ( max < input ) {
  max = input;
}
```

where `max` and `input` are the maximum and the input, respectively; ditto for the minimum

This operation is valid for the second input number on

Policy

- Process the first number differently than the rest

Policy

- Process the first number differently than the rest
  - Assign the first number to the total
  - Assign the first number to the maximum and the minimum

Policy

- Process the first number differently than the rest
    - Assign the first number to the total
    - Assign the first number to the maximum and the minimum
- Then receive the rest and perform the updates
    - Add the number to the total
    - Compare the number to the max and then the min for an update

Operation at the Conclusion

Divide the total with the number of data to obtain average, and then print the average, the max, and the min

Operation at the Conclusion

Divide the total with the number of data to obtain average, and then print the average, the max, and the min

Problem might arise if the number of data the user specifies is 0 (dividing by 0)

Operation at the Conclusion

Divide the total with the number of data to obtain average, and then print the average, the max, and the min

Problem might arise if the number of data the user specifies is 0 (dividing by 0)

Terminate the program if the number is less than 1

throw

- Java has a number of runtime error types and they are objects
- To create an error use a statement:

  `throw new <exceptionName>(<message>);`

  Here `<exceptionName>` is the name of the error and `<message>` is the error message to be printed on screen
- The program is terminated after producing the error message

Here we use `IllegalArgumentException`

```
if ( nData <= 0 ) {
  throw new IllegalArgumentException(
     "\n    \#Data must be positive: nData" );
}
```

here `nData` is the number of data that the user enters

A Special Method for Receiving an Input Number

Use a special method `getData` that takes the index to the data and a Scanner as parameters, prompts the user for data at the index, receives the data, and returns the data

The Code

```java
import java.util.Scanner;
public class CumulativeSum {
  public static double getData( int index, Scanner console ) {
    System.out.print( "Enter data #" + index + ": " );
    double input = console.nextDouble();
    return input;
  }
```

Method header

The Code

```java
import java.util.Scanner;
public class CumulativeSum {
  public static double getData( int index, Scanner console ) {
    System.out.print( "Enter data #" + index + ": " );
    double input = console.nextDouble();
    return input;
  }
```

Prompt the user and receive the input

The Code

```java
import java.util.Scanner;
public class CumulativeSum {
  public static double getData( int index, Scanner console ) {
    System.out.print( "Enter data #" + index + ": " );
    double input = console.nextDouble();
    return input;
  }
```

Return the input

Storing the value into `input` can be skipped by chaing it to:

`return console.nextDouble();`

The Code (cont'd)

```
  public static void main( String[] args ) {
    double total, max, min, average, input;
    Scanner console = new Scanner( System.in );
    System.out.print( "Enter the # of data:  " );
    int nData = console.nextInt();
    if ( nData <= 0 ) {
      throw new IllegalArgumentException(
          "#Data must be positive: nData" );
    }
    input = getData( 1, console );
    total = input;
    max = input;
    min = input;
```

Define the double variables; declare and define the console variable

The Code (cont'd)

```java
  public static void main( String[] args ) {
    double total, max, min, average, input;
    Scanner console = new Scanner( System.in );
    System.out.print( "Enter the # of data:  " );
    int nData = console.nextInt();
    if ( nData <= 0 ) {
      throw new IllegalArgumentException(
          "#Data must be positive: nData" );
    }
    input = getData( 1, console );
    total = input;
    max = input;
    min = input;
```

Receive the number of inputs

The Code (cont'd)

```
  public static void main( String[] args ) {
    double total, max, min, average, input;
    Scanner console = new Scanner( System.in );
    System.out.print( "Enter the # of data: " );
    int nData = console.nextInt();
    if ( nData <= 0 ) {
      throw new IllegalArgumentException(
          "#Data must be positive: nData" );
    }
    input = getData( 1, console );
    total = input;
    max = input;
    min = input;
```

Number check

The Code (cont’d)

```
  public static void main( String[] args ) {
    double total, max, min, average, input;
    Scanner console = new Scanner( System.in );
    System.out.print( "Enter the # of data: " );
    int nData = console.nextInt();
    if ( nData <= 0 ) {
      throw new IllegalArgumentException(
          "#Data must be positive: nData" );
    }
    input = getData( 1, console );
    total = input;
    max = input;
    min = input;
```

Receive the first number

The Code (cont'd)

```java
  public static void main( String[] args ) {
    double total, max, min, average, input;
    Scanner console = new Scanner( System.in );
    System.out.print( "Enter the # of data: " );
    int nData = console.nextInt();
    if ( nData <= 0 ) {
      throw new IllegalArgumentException(
          "#Data must be positive: nData" );
    }
    input = getData( 1, console );
    total = input;
    max = input;
    min = input;
```

Process the first number

The Code (cont'd)

```
    for ( int index = 2; index <= nData; index ++ ) {
      input = getData( index, console );
      total += input;
      if ( max < input ) {
        max = input;
      }
      else if ( min > input ) {
        min = input;
      }
    }
    average = total / nData;
    System.out.println( "Total:          " + total );
    System.out.println( "Average:        " + average );
    System.out.println( "Maximum:        " + max );
    System.out.println( "Minimum:        " + min );
  }
}
```

Loop for the second number on

The Code (cont’d)

```
    for ( int index = 2; index <= nData; index ++ ) {
      input = getData( index, console );
      total += input;
      if ( max < input ) {
        max = input;
      }
      else if ( min > input ) {
        min = input;
      }
    }
    average = total / nData;
    System.out.println( "Total:         " + total );
    System.out.println( "Average:       " + average );
    System.out.println( "Maximum:       " + max );
    System.out.println( "Minimum:       " + min );
  }
}
```

Receive the number

The Code (cont'd)

```
    for ( int index = 2; index <= nData; index ++ ) {
      input = getData( index, console );
      total += input;
      if ( max < input ) {
        max = input;
      }
      else if ( min > input ) {
        min = input;
      }
    }
    average = total / nData;
    System.out.println( "Total:          " + total );
    System.out.println( "Average:        " + average );
    System.out.println( "Maximum:        " + max );
    System.out.println( "Minimum:        " + min );
  }
}
```

Update the total

The Code (cont’d)

```
    for ( int index = 2; index <= nData; index ++ ) {
      input = getData( index, console );
      total += input;
      if ( max < input ) {
        max = input;
      }
      else if ( min > input ) {
        min = input;
      }
    }
    average = total / nData;
    System.out.println( "Total:          " + total );
    System.out.println( "Average:        " + average );
    System.out.println( "Maximum:        " + max );
    System.out.println( "Minimum:        " + min );
  }
}
```

Update the maximum and the minimum

The Code (cont’d)

```
    for ( int index = 2; index <= nData; index ++ ) {
      input = getData( index, console );
      total += input;
      if ( max < input ) {
        max = input;
      }
      else if ( min > input ) {
        min = input;
      }
    }
    average = total / nData;
    System.out.println( "Total:          " + total );
    System.out.println( "Average:        " + average );
    System.out.println( "Maximum:        " + max );
    System.out.println( "Minimum:        " + min );
  }
}
```

Compute the average

The Code (cont'd)

```java
    for ( int index = 2; index <= nData; index ++ ) {
      input = getData( index, console );
      total += input;
      if ( max < input ) {
        max = input;
      }
      else if ( min > input ) {
        min = input;
      }
    }
    average = total / nData;
    System.out.println( "Total:          " + total );
    System.out.println( "Average:        " + average );
    System.out.println( "Maximum:        " + max );
    System.out.println( "Minimum:        " + min );
  }
}
```

Output the results

The End