Topics: Selection (conditional) statement, input using JLiveRead class, while loop

Reading: (GG) Sec 2.3.3; (PL) Lesson page 1-4, 1-5

Example 1: Temperature conversion, re-visited

Write a program to convert a user-entered temperature in degrees Fahrenheit to degrees Celsius. Display the temperature both in °F and °C.

// Convert temperature from F to C

class TempConvert {
    public static void main(String[] args) {
        final int FREEZE = 0; // freezing point

        System.out.print("Enter temp in F: ");
        double tempF = JLiveRead.readDouble();
        double tempC;
        tempC = (tempF - 32) * 5 / 9;
        System.out.println(tempF + " F is " + tempC + " C");

        if (tempC < FREEZE)
            System.out.println("So very cold!");
        else
            System.out.println("Tolerable");
    }
}

User Input

We'll use methods in class JLiveRead to read in user input from the keyboard. See ProgramLive Lesson page 1-5 for more information on input and on the JLiveRead class.

Examples:
    int var1 = JLiveRead.readInt();
    double var2 = JLiveRead.readDouble();
    char var3 = JLiveRead.readChar();
    boolean var4 = JLiveRead.readBoolean();

Shortcut expressions

Increment: i++;
Decrement: i--;

Assignment operators:
    s += val;
    s -= val;
    s *= val;
    s /= val;
Conditional Statement

\[
\begin{align*}
\text{if ( condition1 )} & \quad \text{statement1;} \\
\text{else if ( condition2 )} & \quad \text{statement2;} \\
\text{else} & \quad \text{statement3;} \\
\end{align*}
\]

The while loop

\[
\begin{align*}
\text{while ( condition )} & \quad \text{statement ;} \\
\text{int i= 1;} & \quad \text{while ( i<=n )} \& \{ \\
& \quad \text{// do something} \\
& \quad \text{// increment counter} \\
& \quad \text{i= i + 1;} \\
& \} \\
\end{align*}
\]

Pattern for doing something \( n \) times

\[
\begin{align*}
\text{Pattern for doing something an indefinite number of times} \\
\text{% initialization} \\
\text{while ( not stopping signal )} \& \{ \\
& \quad \text{// do something} \\
& \quad \text{// update status (variables)} \\
& \}
\end{align*}
\]

Example 2: Shipping cost

A shipping company calculates shipping prices as follows:

- A package weighing 5 lbs (pounds) or less costs $12, excluding tax.
- A package weighing over 5 lbs and less than 10 lbs costs $18, excluding tax.
- A package weighing at least 10 lbs costs $20 plus $1.50 for each pound over 10 lbs. For example, a 10.5 lb package costs $20.75 to ship, excluding tax.
- Tax (8%) is charged for shipment to Region 1. No tax is charged for shipment to Region 2.

Write a program segment that

- prompts the user for package weight (variable \( \text{wt} \)), destination code (variable \( \text{code} \)). Code will be entered as 1 or 2.
- calculates the shipping charge (including tax) and stores the result in variable \( \text{charge} \)
- prints the value of \( \text{charge} \) in a descriptive sentence.

```java
double charge;  // shipping charge

// Get the package weight and destination code
System.out.print("Enter the package weight: ");
double weight= JLiveRead.readDouble();
System.out.print("Enter destination code (either 1 or 2): ");
int des_code= JLiveRead.readInt();

//continue on next page...
```
/ Calculate shipping charge without tax

// Add tax for destination code 1

System.out.println("The shipping charge (including tax) = $" + charge);

Example 3: Eeeeeeeeeee!

The exponential function can be approximated by the series \( e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} \). One expects that the approximation is “better” when more terms in the series are used.

We will use method \( \text{Math.exp()} \) to calculate the “true” value of \( e^x \) and attempt to determine “how good” the above series approximation is. The difference between the true value and the approximation is the \textit{error}. When we approximate, the amount of error that we are willing to tolerate is called the \textit{tolerance}.

Write a program segment that starts by approximating \( e^x \) by just the first term of the series and then add one term at a time until a tolerance of 0.0001 is satisfied. \( x \) is to be input by a user.

```java
System.out.print("Enter power of e: ");
double x = JLiveRead.readDouble();
double ans = Math.exp(x); // true value of e^x
double ex = 1; // approx value of e^x so far
double tol = 0.0001; // error tolerance

System.out.print("Error after ");
System.out.println(" terms: ");
```

```java
System.out.println(" + Math.abs(ans-ex));
```