Topics: Selection (conditional) statement, keyboard input using Scanner class, while loop
Reading: Sec 2.2, Sec 3.1 (exclude pp104-109), Sec 3.2 (exclude pp112-114), pp 130-135 of Sec 3.3 (exclude the do-while loop)

Example: Quadratic function, re-visited

Write a program to find the minimum value of the quadratic function \( q(x) = x^2 + bx + c \) on the interval \([L, R]\).

/* Min value of \( q(x) = x^2 + bx + c \) on interval \([L,R]\) */
public class MinQuadratic {
    public static void main(String[] args) {
        final double b=2, c=-1.5;
        double L=-3, R=5;
        double qMin, qL, qR;  // Min value of q, q(L), q(R)
        double xc= -b/2;
        if (L<=xc && xc<=R)  // qMin is q(xc)
            qMin= xc*xc + b*xc + c;
        else {
            // qMin is q(L) or q(R)
            qL= L*L + b*L + c;
            qR= R*R + b*R + c;
            if (qL < qR)
                qMin= qL;
            else
                qMin= qR;
        }
        System.out.println("Min value is " + qMin);
    }
}

User Input

We’ll use the class Scanner to read in user input from the keyboard. First, you need to import the class using the import statement outside of the class body:

    import java.util.Scanner;

Inside a method (e.g., main method), you create an object of the Scanner class. Below, we create such an object and refer to it with the variable keyboard:

    Scanner keyboard= new Scanner(System.in);

Now we can use keyboard to read user input. Below are some example method calls. Read Sec 2.2 (Savitch) for more information on the Scanner class.

Examples: int var1= keyboard.nextInt();
           double var2= keyboard.nextDouble();
           char var3= keyboard.nextChar();
           boolean var4= keyboard.nextBoolean();
Shortcut expressions

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

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

Conditional Statement

if ( condition1 )
    statement1;

if ( condition1 )
    statement1;
else
    statement2;

if ( condition1 )
    statement1;
else if ( condition2 )
    statement2;
else
    statement3;

The while loop

while ( condition )
    statement;

Pattern for doing something $n$ times

int i= 1;
while ( i<=n ) {
    // do something
    // increment counter
    i= i + 1;
}

Pattern for doing something an indefinite number of times

% initialization
while ( not stopping signal ) {
    // do something
    // update status (variables)
}

Example: Factorial

Write a program fragment to calculate $k!$ (the factorial of $k$). Assume $k$ is given and $k \geq 0$. Use a while loop.
Example: 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 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 error. When we approximate, the amount of error that we are willing to tolerate is called the tolerance.

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

```java
import java.util.Scanner;

/* Approximate e^x using series 1 + x/1! + (x^2)/2! + ... */
public class Eeee {
    public static void main(String[] args) {
        Scanner keyboard= new Scanner(System.in);

        System.out.print("Enter power of e: ");
        double x= keyboard.nextDouble();
        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
        int kfact;                // k!
        int i, k;

        System.out.print("Error after " + k);
        System.out.println(" terms:  " + Math.abs(ans-ex));
    }
}
```