Testing Polynomials

Tutorial 7 (10th March 2021)

$$3x^4 - 4x^3 + 2x - 8$$

Last week

```
public static int readIntWithPrompt( String prompt ) {
    Scanner in = new Scanner( System.in );
    while( true ) {
        System.out.print(prompt);
        try {
            return in.nextInt();
        } catch( InputMismatchException e ) {
            in.nextLine();
        }
    }
}
```


```
public static int readIntWithPrompt( String prompt ) {
    Scanner scan = new Scanner( System.in );
    while ( true ) {
        System.out.print( prompt );
        if ( scan.hasNextInt() ) {
            return scan.nextInt();
        } else {
            System.out.println("Integer expected: " + scan.nextLine());
        }
    }
}
```

Last week (2): leaves as enum

```
public interface QTNode {
    public int countBlacks( int size );
}
```

```
public class WhiteLeaf implements QTNode {
    @Override
    public int countBlacks( int size ) {
        return 0;
    }
}
public class BlackLeaf implements QTNode {
    @Override
    public int countBlacks( int size ) {
        return size * size;
    }
}
```

- Only one instance is needed of both class `BlackLeaf` and `WhiteLeaf`.
- We can use the *singleton pattern*.

```
public enum Leaf implements QTNode {
    BlackLeaf( false ), WhiteLeaf( true );
    private final boolean isWhite;

    private Leaf( boolean isWhite ) {
        this.isWhite = isWhite;
    }

    @Override
    public int countBlacks( int size ) {
        return isWhite ? 0 : size * size ;
    }
}
```

$$3x^4 - 4x^3 + 2x - 8$$

Donald Ervin Knuth (1938 - …)

- author of (the multi-volume work) The Art of Computer Programming
- creator of the TeX computer typesetting system

“Beware of bugs in the above code; I have only proved it correct, not tried it.”

Terms

$$3x^4 - 4x^3 + 2x - 8$$

```
public class Term {

    /**
     * Each term consists of a coefficient and an exponent
     */
    private double coefficient;
    private int exponent;
    …
}
```

Polynomials

```
public class Polynomial {

    /**
     * A polynomial is a sequence of terms here kept in an List
     */
    List<Term> terms;

    /**
     * A constructor for creating the zero Polynomial.
     * zero is presented as an empty list of terms and not as a
     * single term with 0 as a coefficient
     */
    public Polynomial() {
        terms = new LinkedList<>();
    }
```

Polynomials: copy constructor

```
public Polynomial( Polynomial p ) {
    terms = new LinkedList<>();
    for ( Term t : p.terms ) {
        terms.add( new Term(t) );
    }
}
```

Polynomials: addition

All operations are implemented destructively

- `this` object is updated with result

$$(1 + 2x + 3x^2) + (2 + x + 4x^3)$$
$$= 3 + 3\mathrm{x} + 3x^2 + 4x^3$$

Use (list)iterators to traverse list of terms

Order terms according to their exponent

Complexity: **O**(*N*) (*N* = #terms)

Polynomials: addition

```
public void plus( Polynomial b ) {
    ListIterator<Term> lita = terms.listIterator(),
                       litb = b.terms.listIterator();
    while ( lita.hasNext() && litb.hasNext() ) {
        Term ta = lita.next(), tb = litb.next();
        if ( ta.getExp() == tb.getExp() ) {
            ta.plus(tb);
            if ( ta.getCoef() == 0 ) {
                lita.remove();
            }
        } else if (ta.getExp() > tb.getExp()) {
            …
        } …
    }
    …
}
```

Polynomials: multiplication

$$(1 + 2x + 3x^2) \cdot (2 + x + 4x^2)$$

$$= 2 + 5x + 12x^2 + 11x^3 + 12x^4$$

Complexity: **O**(N^2) (N = #terms)

Introduce helper methods:

```
private Polynomial timesTerm( Term t ) {
    for ( Term rt : terms ) {
        rt.times(t);
    }
    return this;
}

private static Polynomial polyTimesTerm( Polynomial p, Term t ) {
    return ( new Polynomial( p ) ).timesTerm( t );
}
```

Polynomials: division

When you divide two polynomials you can check the answer using the following:

$$\text{dividend} = (\text{quotient} \cdot \text{divisor}) + \text{remainder}$$

The result is written in the form:

$$\text{dividend} \div \text{divisor} = \text{quotient} + \frac{\text{remainder}}{\text{divisor}}$$

Example: Divide $x^2 + 3x - 2$ by $x + 1$ and check the answer.

$$
\begin{array}{r}
x + 2 \\
x+1 \overline{)\, x^2 + 3x - 2} \\
\underline{x^2 + x} \\
2x - 2 \\
\underline{2x + 2} \\
-4
\end{array}
$$

remainder

1. $x \overline{)\, x^2} = \dfrac{x^2}{x} = x$
2. $x(x+1) = x^2 + x$
3. $(x^2 + 3x) - (x^2 + x) = 2x$
4. $x \overline{)\, 2x} = \dfrac{2x}{x} = 2$
5. $2(x+1) = 2x + 2$
6. $(2x - 2) - (2x + 2) = -4$

Answer: $x + 2 + \dfrac{-4}{x+1}$

Check: $\underbrace{(x+2)}_{\text{quotient}}\underbrace{(x+1)}_{\text{divisor}} + \underbrace{(-4)}_{\text{remainder}} = \underbrace{x^2 + 3x - 2}_{\text{dividend}}$ correct

Example: Divide $2x^3 + 4x - 1$ by $2x - 2$ and check the answer.

$$
\begin{array}{r}
x^2 + x + 3 \\
2x-2\,\overline{)\,2x^3 + 0x^2 + 4x - 1} \\
\underline{2x^3 - 2x^2} \\
2x^2 + 4x \\
\underline{2x^2 - 2x} \\
6x - 1 \\
\underline{6x - 6} \\
5
\end{array}
$$

Write the terms of the dividend in descending order.

Since there is no x^2 term in the dividend, add $0x^2$ as a placeholder.

1. $\frac{2x^3}{2x} = x^2$
2. $x^2(2x-2) = 2x^3 - 2x^2$
3. $2x^3 - (2x^3 - 2x^2) = 2x^2$
4. $\frac{2x^2}{2x} = x$
5. $x(2x-2) = 2x^2 - 2x$
6. $(2x^2 + 4x) - (2x^2 - 2x) = 6x$
7. $\frac{6x}{2x} = 3$
8. $3(2x-2) = 6x - 6$
9. $(6x-1) - (6x-6) = 5 = \text{remainder}$

Answer: $x^2 + x + 3 + \dfrac{5}{2x-2}$

Check: $(x^2 + x + 3)(2x - 2) + 5$

$= 2x^3 + 4x - 1$

Unit testing

Many useful properties

- commutativity: a + b = b + a
- associativity: (a + b) + c = a + (b + c)
- distributivity: a * (b + c) = a*b + a*c
- additive identity: a + 0 = 0 + a = a
- multiplicative identity: a * 1 = 1 * a = a

Example: associativity of +

```
// Associativity using only positive polynomials.
// (1.0*x^1 + 2.0*x^2) + 3.0*x^3 = 1.0*x^1 + (2.0*x^2 + 3.0*x^3)
Polynomial p13 = new Polynomial( "1.0 1" ),
           p14 = new Polynomial( "2.0 2" ),
           p15 = new Polynomial( "3.0 3" );

Polynomial p16 = new Polynomial( p13 ),
           p17 = new Polynomial( p14 );
           p18 = new Polynomial( p15 );

p13.plus( p14 );
p13.plus( p15 );

p17.plus( p18 );
p16.plus( p17 );

assertEquals( "(1.0*x^1 + 2.0*x^2) + 3.0*x^3 = 1.0*x^1 + (2.0*x^2 + 3.0*x^3)",
p13, p16 );
```

Finally