Disclaimer: These notes may be distributed outside this class only with the permission of the Instructor.

1.1 Course Introduction

Check the course website for the syllabus and course introduction slides.

1.2 Java Review

We start by introducing Object Oriented Programming (OOP) concepts, such as class, inheritance, and encapsulation.

1.2.1 Code Examples: Fraction Class

In this example, we implement a Fraction class, which can represent fractions and support arithmetic operations on the fractions. A common fraction consists of an integer numerator, and non-zero denominator. For example: \( \frac{2}{10} \) or \( \frac{13}{5} \). The Fraction class has two private members numerator and denominator.

Listing 1: Fraction Class

```java
/**
 * Fraction class implements non-negative fractions
 * @author anwar
 */
public class Fraction {
    protected int numerator;
    protected int denominator;
    /** Constructs a Fraction n/d.
     * @param n is the numerator, assumed non-negative.
     * @param d is the denominator, assumed positive.
     */
    Fraction(int n, int d) {
        int g = gcd(d, n);
        /** reduce the fraction */
        numerator = n/g;
        denominator = d/g;
    }
    /** Constructs a Fraction n/1.
     * @param n is the numerator, assumed non-negative.
     */
    *
```
public Fraction(int n) {
    this(n, 1);
}

/** Constructs a Fraction 0/1. */
public Fraction() {
    numerator = 0;
    denominator = 1;
}

public String toString() {
    return (numerator + "/" + denominator);
}

/** Calculates and returns the double floating point value of a fraction. 
 * @return a double floating point value for this Fraction. */
public double evaluate() {
    double n = numerator;  // convert to double
    double d = denominator;
    return (n / d);
}

/** Add f2 to this fraction and return the result. 
 * @param f2 is the fraction to be added. 
 * @return the result of adding f2 to this Fraction. */
public Fraction add(Fraction f2) {
    Fraction r = new Fraction((numerator * f2.denominator) +
                              (f2.numerator * denominator),
                              (denominator * f2.denominator));
    return r;
}

/** Subtract f2 from this fraction and return the result. 
 * @param f2 is the fraction to be added. 
 * @return the result of adding f2 to this Fraction. */
public Fraction sub(Fraction f2) {
    Fraction r = new Fraction((numerator * f2.denominator) -
                              (f2.numerator * denominator),
                              (denominator * f2.denominator));
    return r;
}

/** Multiply f2 to this fraction and return the result. 
 * @param f2 is the fraction to be added. 
 * @return the result of adding f2 to this Fraction. */
public Fraction mul(Fraction f2) {

public Fraction div(Fraction f2) {
    return (new Fraction(numerator * f2.denominator, 
        denominator * f2.numerator));
}

/** Computes the greatest common divisor (gcd) of the two inputs. 
* @param a is assumed positive 
* @param b is assumed non-negative 
* @return the gcd of a and b 
*/
static private int gcd(int a, int b) {
    int t;
    // a must be greater than or equal to b
    if (a < b) {
        t = a;
        a = b;
        b = t;
    }
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}

public static void main(String[] argv) {
    /* Test all three constructors and toString. */
    Fraction f0 = new Fraction();
    Fraction f1 = new Fraction(3);
    Fraction f2 = new Fraction(12, 20);
    System.out.println("\nTesting constructors (and toString):");
    System.out.println("The fraction f0 is " + f0.toString());
    System.out.println("The fraction f1 is " + f1); // toString is implicit
    System.out.println("The fraction f2 is " + f2);
    /* Test methods on Fraction: add and evaluate. */
    System.out.println("\nTesting add and evaluate:");
    System.out.println("The floating point value of " + f1 + " is " +
        f1.toString());
}
Lecture 1:

```java
f1.evaluate());
System.out.println("The floating point value of \f2 is "+
f2.evaluate());

Fraction sumOfTwo = f1.add(f2);
Fraction sumOfThree = f0.add(f1.add(f2));

System.out.println("The sum of \f1 and \f2 is "+sumOfTwo);
System.out.println("The sum of \f0, \f1 and \f2 is "+sumOfThree);

/**
 * test sub, div, mul here
 */

/* Test gcd function (static method). */
System.out.println("\nTesting gcd:");
System.out.println("The gcd of 2 and 10 is: "+gcd(2, 10));
System.out.println("The gcd of 15 and 5 is: "+gcd(15, 5));
System.out.println("The gcd of 24 and 18 is: "+gcd(24, 18));
System.out.println("The gcd of 10 and 10 is: "+gcd(10, 10));
System.out.println("The gcd of 21 and 400 is: "+gcd(21, 400));
}
```

MixedFraction inherits Fraction, so that it inherits addition, subtraction etc. Only difference is that mixed fraction is a whole number and a fraction combined. For example: \(1 \frac{3}{5} = \frac{13}{5}\)

Listing 2: MixedFraction Class

```java
/**
 * This class implements mixed fraction
 * @author anwar mamat
 */
public class MixedFraction extends Fraction{
    /** Constructs a Fraction m n/d. 
     * @param m is the integer part.
     * @param n is the numerator, assumed non-negative.
     * @param d is the denominator, assumed positive.
     */
    public MixedFraction(int m, int n, int d){
        super(m*d+n, d);//convert mixed fraction into proper fraction.
    }

    /** Constructs a Fraction m n/d. 
     * @param f is a fraction
     */
    public MixedFraction(Fraction f) {
        super(f.numerator, f.denominator);
    }

    public String toString() {
        int m = numerator / denominator;
        int n = numerator % denominator;
```
Main is the fraction test driver class. The "main" method of this class creates Fraction objects and prints the result in the console.

Listing 3: Fraction Test Driver

```java
public class Main{
    public static void main(String[] args) {
        Fraction f1 = new Fraction(2,10);
        Fraction f2 = new MixedFraction(1,2,10);
        System.out.println("f1=" + f1);
        System.out.println("f2=" + f2);
        MixedFraction f3 = new MixedFraction(f2.add(f1));
        System.out.println(f1 + " + " + f2 + " = " + f3);
        Fraction f4 = f2.sub(f1);
        System.out.println(f2 + " - " + f1 + " = " + f4);
        Fraction f5 = f1.mul(f2);
        System.out.println(f1 + " * " + f2 + " = " + f5);
        Fraction f6 = f1.div(f2);
        System.out.println(f1 + " / " + f2 + " = " + f6);
    }
}
```

We also create a JUnit test class for fraction.

Listing 4: Fraction JUnit Test

```java
import static org.junit.Assert.*;
import org.junit.Test;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

public class FractionTest {
    
    public FractionTest() {
    }
    
    @BeforeClass
    public static void setUpClass() {
        System.out.println("@UtilsJUnit4Test: @BeforeClass: method");
    }
```
@AfterClass
public static void tearDownClass() {
    System.out.println("*UtilsJUnit4Test: @tearDownClass method");
}

@Before
public void setUpAgain() {
    System.out.println("*UtilsJUnit4Test: @setUp method");
}

@After
public void tearDown() {
    System.out.println("*UtilsJUnit4Test: @tearDown method");
}

/**
 * Test of toString method, of class Fraction.
 */
@test
public void testToString() {
    System.out.println("toString");
    Fraction instance = new Fraction(2,10);
    String expResult = "1/5";
    String result = instance.toString();
    assertEquals(expResult, result);
}

/**
 * Test of evaluate method, of class Fraction.
 */
@test
public void testEvaluate() {
    System.out.println("evaluate");
    Fraction instance = new Fraction(5,10);
    double expResult = 0.5;
    double result = instance.evaluate();
    assertEquals(expResult, result, 0.0);
}

/**
 * Test of add method, of class Fraction.
 */
@test
public void testAdd() {
    System.out.println("add");
    Fraction f2 = new Fraction(2,7);
    Fraction instance = new Fraction(1,5);
    Fraction expResult = new Fraction(17,35);
    Fraction result = instance.add(f2);
    assertEquals(expResult, result);
}
    /**
     * Test of sub method, of class Fraction.
     */
    @Test
    public void testSub() {
      System.out.println("sub");
      Fraction f2 = new Fraction(1,5);
      Fraction instance = new Fraction(4,10);
      Fraction expResult = new Fraction(1,5);
      Fraction result = instance.sub(f2);
      assertEquals(expResult, result);
    }
    
    /**
     * Test of mul method, of class Fraction.
     */
    @Test
    public void testMul() {
      System.out.println("mul");
      Fraction f2 = new Fraction(3,5);
      Fraction instance = new Fraction(2,3);
      Fraction expResult = new Fraction(6,15);
      Fraction result = instance.mul(f2);
      assertEquals(expResult, result);
    }
    
    /**
     * Test of div method, of class Fraction.
     */
    @Test
    public void testDiv() {
      System.out.println("div");
      Fraction f2 = new Fraction(2,5);
      Fraction instance = new Fraction(3,7);
      Fraction expResult = new Fraction(15,14);
      Fraction result = instance.div(f2);
      assertEquals(expResult, result);
    }
  }