Assignment 1:

(Joseph Sakkab, Adnan Ifram)
(20880535, 20881975)
(jsakkab@uwaterloo.ca, aifram@uwaterloo.ca)

Part 1:

Test plan:

Our approach for testing the methods will be testing normal, unusual, and extreme cases for each of the methods, first we will begin by creating normal cases with simple numbers and common equations, then create unusual cases which will be a bit harder but still manageable, and finally creating extreme cases which will test the accuracy and implementation of these cases, moreover we will test for errors and exceptions by creating a method for each value that would return either a NullPointerException or IllegalArgumentException.

Tests:

---

public Polynomial (String str)

public void testConstructorDefault() {
Creating a new default polynomial

- Input: Polynomial p = new Polynomial();
- Output: "0.0"
- Characteristics: degree -1, value 0.0

Default degree = -1

- Input: new Polynomial().getDegree();
- Output: "-1"
- Characteristics:

Default value must equal 0

- Input: new Polynomial().getValue(0);
- Output: "0"
- Characteristics:

Default value must equal 0

- Input: new Polynomial().getValue(5.0);
- Output: "0"
- Characteristics:

Printing default polynomial

- Input: new Polynomial().toString();
- Output: “”
- Characteristics:

Default value also must equal 0.0

- Input: new Polynomial().getValue(0);
- Output: “”
- Characteristics:

Default value also must equal 0.0

- Input: new Polynomial().getValue(33);
- Output: “”
- Characteristics:

Default value also must equal 0.0

- Input: new Polynomial().getValue(-500.0);
- Output: “”
- Characteristics:

---

public void testConstructorString() {

creating null polynomial = 0

- Input: new Polynomial(null).toString()
- Output: “”
- Characteristics:

Sort by degree:

- Input: Polynomial pHighestToLowest = new Polynomial("-3.2x^5 + 5x^6 + 80.0");
- Output: "5.0x^6 - 3.2x^5 + 80.0"
- Characteristics: degree 6, arranged in order

Converts int values to double:

- Input: Polynomial pIntegersOnly = new Polynomial("-3x^2 + 5");
- Output: “-3.0x^2 + 5.0”
- Characteristics: degree 2, all converted to doubles

Allows large decimals:

- Input: Polynomial pDecimals = new Polynomial("-3.6789x^2 + 5.20394");

- Output: “-3.6789x^2 + 5.20394”
- Characteristics: degree 2, all decimals are included

P(x) = 0.0
- Input: Polynomial pDefault = new Polynomial("0.0");
- Output: “0”
- Characteristics: the polynomial is the integer and not the double 0

x^0 converted to 1
- Input: Polynomial pZeroDegree = new Polynomial("5.0x^0");
- Output: “5.0”
- Characteristics: removes the x^0 as it equates to 1.0

x^1 converted to x
- Input: Polynomial pFirstDegree = new Polynomial("1.3x^1");
- Output: “1.3x”
- Characteristics: removes the power as it is unnecessary

x without power
- Input: new Polynomial("3.0x^2 - x + 1.0")
- Output: “3.0x^2 - x + 1.0”
- Characteristics: recognizes that the x has a coefficient of 1 and power of 1

Positive with no coefficients
- Input: new Polynomial("1x^2 + 1.0x");
- Output: “”
- Characteristics:

Negative with no coefficients
- Input: new Polynomial("-1x^2 - 1.0x");
- Output: “”
- Characteristics:

Negative coefficients
- Input: new Polynomial("-3.0x^3 + -2.0x + -5.0")
- Output: “”
- Characteristics:

Large spaces
- Input: new Polynomial("3.0x^2 - 2.0x   +   1.0")
- Output: “"3.0x^2 - 2.0x + 1.0"”

- Characteristics:

No + or -

- Input: new Polynomial("3.0x^2 2.0x 1.0")
- Output: “"3.0x^2 + 2.0x + 1.0"”
- Characteristics:

Large coefficients

- Input: new Polynomial("500000000.00x^6 + 7.8x^3 + 9.0")
- Output: “5.0E8x^6 + 7.8x^3 + 9.0”
- Characteristics:

Large powers

- Input: new Polynomial("5.0x^50034 + 6.5x^32093")
- Output: “5.0x^50034 + 6.5x^32093”
- Characteristics:

Large coefficients and powers

- Input: new Polynomial("500000000.00x^50034 + 654321x^32093")
- Output: “5.0E8x^50034 + 654321.0x^32093”
- Characteristics:

Large negative coefficients and powers

- Input: new Polynomial("-500000000.00x^50034 - 654321x^32093")
- Output: “-5.0E8x^50034 - 654321.0x^32093”
- Characteristics:

Large number with a lot of random numbers

- Input: new Polynomial("-123456789.00x^50034 - 654321x^32093")
- Output: “-1.23456789E8x^50034 - 654321.0x^32093”
- Characteristics:

Adding + in beginning of polynomial

- Input: new Polynomial("+33.0x^5 - 290.5x^3 + 75.9")
- Output: “33.0x^5 - 290.5x^3 + 75.9”
- Characteristics:

Adding – then +coefficient

- Input: new Polynomial("33.0x^5 - +290.5x^3 + -75.9")
- Output: “33.0x^5 - 290.5x^3 - 75.9”

- Characteristics:

Constant

- Input: new Polynomial("512.0")
- Output: "512.0"
- Characteristics:

Negative constant

- Input: new Polynomial("-512.0")
- Output: "-512.0"
- Characteristics:

---

public void Incorrectformat1() {
no white spaces

- Input: new Polynomial("3.0x^2-2.0x+1.0");
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void Incorrectformat2() {
negative power

- Input: new Polynomial("3.0x^-2 + 1.0")
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void testempty() {
creating an empty polynomial

- Input: new Polynomial("");
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void testLetter() {
polynomial with letters

- Input: new Polynomial("HelloWorld!");
- Output: "expected = IllegalArgumentException"

- Characteristics:

---

public void testDecimalDegree() {
double degree

- Input: new Polynomial("3.0x^4.56");
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void testNonNegativeIntegerDegree() {
negative integer degree

- Input: new Polynomial("3.0x^-0.5");
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void testGetDegree() {
polynomial with degree of 2

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").getDegree
- Output: "2"
- Characteristics:

Polynomial with degree of 0

- Input: new Polynomial("5.0x^0");
- Output: "0"
- Characteristics:

---

public void testGetCoefficient() {
coefficient of the second degree getting positive

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").getCoefficient(2);
- Output: "3.0"
- Characteristics:

Coefficient of the first degree getting negative

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").getCoefficient(1);
- Output: "-2.0"

- Characteristics:

Coefficient of the 0 degree

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").getCoefficient(0);
- Output: "1.0"
- Characteristics:

Coefficient of a non existant degree

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").getCoefficient(5);
- Output: "0.0"
- Characteristics:

---

public void testGetNegativeDegree() {
getting coefficient of a negative degree

- Input: p.getCoefficient(-3)
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void testUpdate() {
making sure that it is a mutator not an accessor

- Input: Polynomial p=new Polynomial("3.0x^2 - 2.0x + 1.0")
  p.update(4.0, 2);
  p.toString
- Output: "False"
- Characteristics:

Updating with an existing term

- Input: p.toString
- Output: "7.0x^2 - 2.0x + 1.0"
- Characteristics:

Updating by a 0 coefficient with an existing power

- Input: p.update(0, 2)
- Output: "7.0x^2 - 2.0x + 1.0"
- Characteristics:

Updating by a 0.0 coefficient with a non existing power

- Input: p.update(0.0, 6)
- Output: "7.0x^2 - 2.0x + 1.0"

- Characteristics:

Updating with a negative coefficient

- Input: Polynomial p2 = new Polynomial("3.0x^2 - 2.0x + 1.0")
  p2.update(-3.2, 1)
- Output: "3.0x^2 - 5.2x + 1.0"
- Characteristics:

Updating with a non existing power

- Input: p2.update(5.7, 7)
- Output: "5.7x^7 + 3.0x^2 - 5.2x + 1.0"
- Characteristics:

---

public void testUpdateNegative() {
updating with negative power

- Input: p.update(3.0, -9)
- Output: "expected = IllegalArgumentException"
- Characteristics:

---

public void testAdd() {
checking it is an accessor not a mutator

- Input: Polynomial p1 = p.add(new Polynomial("-3x^2 + 5"))
  p.toString
- Output: "3.0x^2 - 2.0x + 1.0" whish is p1
- Characteristics:

Adding a non arranged polynomial

- Input: p1.toString
- Output: "5.0x^6 - 3.2x^5 + 3.0x^2 - 2.0x + 81.0"
- Characteristics:

Deletes a 0 coefficient

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").add(new Polynomial("-3.0x^2 - 2.0x + 1.0"));
- Output: "-4.0x + 2.0"
- Characteristics:

Adding a negated polynomial

- Input: new Polynomial("-3.0x^2 - 2.0x + 1.0").add(new Polynomial("3.0x^2 + 2.0x - 1.0"));
- Output: "0"
- Characteristics:

Adding default polynomial

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").add(new Polynomial("0.0"));
- Output: "3.0x^2 - 2.0x + 1.0"
- Characteristics:

---

public void testSubtract() {
checking it is an accessor not a mutator

- Input: p1= p.subtract(pHighestToLowest)
  p.toString;
- Output: "3.0x^2 - 2.0x + 1.0" => p
- Characteristics: : subtracting 2 polynomials checking the original polynomials did not change

Subtracting a non arranged polynomial

- Input: p1.toString;
- Output: "-5.0x^6 + 3.2x^5 + 3.0x^2 - 2.0x - 79.0"
- Characteristics: A new polynomial which is the difference of the polynomials

Deletes a 0 coefficient

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").subtract(new Polynomial("-3.0x^2 - 2.0x + 1.0"));
- Output: "6.0x^2"
- Characteristics: tests the deletion of a term

subtracting itself

- Input: p2.subtract(p2);
- Output: "0"
- Characteristics: subtracting the polynomial to itself returns the zero polynomial

Subtracting default polynomial

- Input: new Polynomial("3.0x^2 - 2.0 x + 1.0").subtract(pDefault);
- Output: "3.0x^2 - 2.0 x + 1.0"

- Characteristics: subtracts a 0 polynomial to the polynomial which leads to no changes

---

public void testNegate() {
testing that it is an accessor not a mutator

- Input: Polynomial pCheck = p.negate();
  p.toString;
- Output: "3.0x^2 - 2.0x + 1.0"
- Characteristics: p should not have changed

Negating a polynomial

- Input: p.negate().toString()
- Output: "-3.0x^2 + 2.0x - 1.0"
- Characteristics: a new polynomial formed by multiplying the old polynomial by -1

Negating the 0 polynomial

- Input: pDefault.negate().toString()
- Output: "0"
- Characteristics: a zero polynomial

---

public void testGetValue() {
getting value when x=1 as double

- Input: p.getValue(1.0)
- Output: "2.0"
- Characteristics: returns the value of replacing the x values in the polynomial by 1.0

Getting value when x=2 as integer

- Input: p.getValu(2)
- Output: "9.0"
- Characteristics: accepts an integer as a parameter and returns a value when substituting 2.0

When value = 0 to a non-null polynomial

- Input: new Polynomial("2.0x^2 - 4.0x").getValue(2.0)
- Output: "0.0"
- Characteristics: returns the correct value when x is replaced with 2 equating to 0

Negative x value

- Input: p1.getValue(-2.0)
- Output: "16.0"
- Characteristics: accept negative integers for the values of x

When the polynomial does not contain any x's for int

- Input: constant.getValue(1))
- Output: "5.6" => constant
- Characteristics: does not change for all values of x

When the polynomial does not contain any x's for large double

- Input: constant.getValue(50.6))
- Output: "5.6" => constant
- Characteristics: does not change for all values of x

When x = 0

- Input: p1.getValue(0.0))
- Output: "0.0"
- Characteristics: substitutes zero in place of x and returns polynomial's value

When x= 0 with constant in the polynomial

- Input: p.getValue(0.0));
- Output: "1.0" = value of constant in the polynomial
- Characteristics: since constant does not have an x so it will remain the same

Getting value of large negative polynomial

- Input: new Polynomial("-500000000.00x^5000 + 654321x^320").getValue(2.0) + "");
- Output: "-Infinity"
- Characteristics: too large of a number lead to the value approaching negative infinity

Getting value of large polynomial

- Input: new Polynomial("500000000.00x^5000 - 654321x^320").getValue(2.0) + "");
- Output: "Infinity"
- Characteristics: too large of a number lead to the value approaching infinity

Fixing the NAN problem for very large negative numbers

- Input: new Polynomial("-500000000.00x^5000 - 654321x^320").getValue(2.0) + "");
- Output: was "NAN" became "-Infinity"
- Characteristics: fixed the not a number problem in teacher's build

Fixing the NAN problem for very large numbers

- Input: new Polynomial("500000000.00x^5000 + 654321x^320").getValue(2.0) + "");
- Output: was "NAN" became "Infinity"
- Characteristics: fixed the not a number problem in teacher's build

---

public void testGetDerivative() {
normal polynomial derivative

- Input: new Polynomial("2.0x^2 - 4.0x").getDerivative
- Output: "2.0x – 4.0"
- Characteristics: deletes the x next to the 4.0 sice it is to the power of 0

removes the x and the carrot and removes the number without an x

- Input: new Polynomial("3.0x^2 - 2.0x + 1.0").getDerivative
- Output: "3.0x^2 – 2.0"
- Characteristics: removes the 1.0 since when deriving the constant has no affect

Removes the power of 1

- Input: new Polynomial("1.3x").getDerivative
- Output: "1.3"
- Characteristics: polynomial becomes a constant

Default polynomial

- Input: new Polynomial("0.0");
- Output: "0.0"
- Characteristics: derivative of 0.0 is 0.0

Derivative of a constant

- Input: new Polynomial("5.0");
- Output: "0.0"
- Characteristics: since a constant does not have a derivative

---

public void testAddNull() {
adding null to a polynomial

- Input: p.add(null)
- Output: "expected = IllegalArgumentException"
- Characteristics: cannot add null

---

public void testSubtractNull() {
subtracting null to a polynomial

- Input: p.subtract(null)
- Output: "expected = IllegalArgumentException"
- Characteristics: cannot subtract null