Marco Martinez
CISP401
12/8/2018
Midterm Extra Credit

CLASS DIAGRAM

Classes

Number
NumberMath
AppDriver

Association

NumberSystem(1) --- includes --- (1) Number
AppDriver(1) --- uses --- (1) NumberMath

Number Class Attributes

INSTANCE VARIABLE DECLARATION
(-) int integerPart
(-) double realPart

CLASS CONSTRUCTORS
(+) Number()
(+) Number(int)
(+) Number(double)
(+) Number(int,double)
(+) Number(float)
(+) Number(int,float)
(+) Number(Number)

CHANGE STATE SERVICES
(+) setInt(int)
(+) setInt(double)
(+) setInt(float)
(+) setReal(int)
(+) setReal(double)
(+) setReal(float)

REAL STATE SERVICES
(+) getInt()
(+) getReal()
(+) toString()

NumberMath Class Attributes

INSTANCE VARIABLE DECLARATION
(-) Number a
(-) Number b
(-) Number result

CLASS CONSTRUCTORS
(+) NumberMath()
(+) NumberMath(int)
(+) NumberMath(double)
(+) NumberMath(int,double)
(+) NumberMath(float)

(+) NumberMath(int,float)
(+) NumberMath(Number)
(+) NumberMath(int,int)
(+) NumberMath(double,double)
(+) NumberMath(int,double,int,double)
(+) NumberMath(float,float)
(+) NumberMath(int,float,int,float)
(+) NumberMath(Number,Number)

CHANGE STATE SERVICES
(+) setA(int)
(+) setA(double)
(+) setA(float)
(+) setA(int,double)
(+) setA(int,float)
(+) setB(int)
(+) setB(double)
(+) setB(float)
(+) setB(int,double)
(+) setB(int,float)
(+) add()
(+) sub()
(+) mul()
(+) div()

REAL STATE SERVICES
(+) getA()
(+) getB()
(+) getResult()
(+) toString()

STATE MODEL

Number Class
Number() → s(null)
Number(int) → s(0)
Number(double) → s(0)
Number(int,double) → s(0)
Number(float) → s(0)
Number(int,float) → s(0)
Number(Number) → s(0)

s(0) → setInt → s(0)
s(0) → setReal → s(0)
s(0) → getInt → s(terminal)
s(0) → getReal → s(terminal)
s(0) → toString → s(terminal)

NumberMath Class

NumberMath() → s(null)
NumberMath(int) → s(0)
NumberMath(double) → s(0)
NumberMath(int,double) → s(0)
NumberMath(float) → s(0)
NumberMath(int,float) → s(0)
NumberMath(Number) → s(0)
NumberMath(int,int) → s(0)
NumberMath(double,double) → s(0)
NumberMath(int,double,int,double) → s(0)
NumberMath(float,float) → s(0)
NumberMath(int,float,int,float) → s(0)
NumberMath(Number,Number) → s(0)

s(0) → setA → s(0)
s(0) → setB → s(0)
s(0) → add → s(0)
s(0) → sub → s(0)
s(0) → mul → s(0)
s(0) → div → s(0)
s(0) → getA → s(terminal)
s(0) → getB → s(terminal)
s(0) → getResult → s(terminal)
s(0) → toString → s(terminal)

USE CASE MODEL (NumberMath)

Normal Scenario 1:
1. User constructs no param object.
2. User exits application.

Normal Scenario 2:
1. User constructs one integer object.
2. User calls add method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 3:
1. User constructs one float object.
2. User calls add method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 4:
1. User constructs one double object.
2. User calls add method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 5:
1. User constructs one integer object.
2. User calls sub method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 6:
1. User constructs one float object.
2. User calls sub method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 7:
1. User constructs one double object.
2. User calls sub method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 8:
1. User constructs one integer object.
2. User calls mul method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 9:
1. User constructs one float object.
2. User calls mul method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 10:
1. User constructs one double object.
2. User calls mul method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 11:
1. User constructs one integer object.
2. User calls div method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 12:
1. User constructs one float object.
2. User calls div method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 13:
1. User constructs one double object.
2. User calls div method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 14:
1. User constructs one integer object.
2. User calls add method.
3. User calls sub method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 15:
1. User constructs one float object.
2. User calls add method.
3. User calls sub method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 16:
1. User constructs one double object.
2. User calls add method.
3. User calls sub method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 17:
1. User constructs one integer object.
2. User calls sub method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 18:
1. User constructs one float object.
2. User calls sub method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 19:
1. User constructs one double object.
2. User calls sub method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 20:
1. User constructs one integer object.
2. User calls add method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 21:
1. User constructs one float object.
2. User calls add method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 22:
1. User constructs one double object.
2. User calls add method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 23:
1. User constructs one integer object.
2. User calls add method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 24:
1. User constructs one float object.
2. User calls add method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 25:
1. User constructs one double object.
2. User calls add method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 26:
1. User constructs one integer object.
2. User calls sub method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 27:
1. User constructs one float object.
2. User calls sub method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 28:
1. User constructs one double object.
2. User calls sub method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 29:
1. User constructs one integer object.
2. User calls mul method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 30:
1. User constructs one float object.
2. User calls mul method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 31:
1. User constructs one double object.
2. User calls mul method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 32:
1. User constructs one integer object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 33:
1. User constructs one float object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 34:
1. User constructs one double object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 35:
1. User constructs one integer object.
2. User calls sub method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 36:
1. User constructs one float object.
2. User calls sub method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 37:
1. User constructs one double object.
2. User calls sub method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 38:
1. User constructs one integer object.
2. User calls add method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 39:
1. User constructs one float object.
2. User calls add method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 40:
1. User constructs one double object.
2. User calls add method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 41:
1. User constructs one integer object.
2. User calls add method.
3. User calls sub method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 42:
1. User constructs one float object.
2. User calls add method.
3. User calls sub method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 43:
1. User constructs one double object.
2. User calls add method.
3. User calls sub method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 44:
1. User constructs one integer object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User calls div method.
6. User requests the processed values via get() method.
7. User exits application.

Normal Scenario 45:
1. User constructs one float object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User calls div method.
6. User requests the processed values via get() method.
7. User exits application.

Normal Scenario 46:
1. User constructs one double object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User calls div method.
6. User requests the processed values via get() method.
7. User exits application.

Normal Scenario 47:
1. User constructs two integer object.
2. User calls add method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 48:
1. User constructs two float object.
2. User calls add method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 49:
1. User constructs two double object.
2. User calls add method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 50:
1. User constructs two integer object.
2. User calls sub method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 51:
1. User constructs two float object.
2. User calls sub method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 52:
1. User constructs two double object.
2. User calls sub method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 53:
1. User constructs two integer object.
2. User calls mul method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 54:
1. User constructs two float object.
2. User calls mul method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 55:
1. User constructs two double object.
2. User calls mul method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 56:
1. User constructs two integer object.
2. User calls div method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 57:
1. User constructs two float object.
2. User calls div method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 58:
1. User constructs two double object.
2. User calls div method.
3. User requests the processed values via get() method.
4. User exits application.

Normal Scenario 59:
1. User constructs two integer object.
2. User calls add method.
3. User calls sub method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 60:
1. User constructs two float object.
2. User calls add method.
3. User calls sub method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 61:
1. User constructs two double object.
2. User calls add method.
3. User calls sub method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 62:
1. User constructs two integer object.
2. User calls sub method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 63:
1. User constructs two float object.
2. User calls sub method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 64:
1. User constructs two double object.
2. User calls sub method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 65:
1. User constructs two integer object.
2. User calls add method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 66:
1. User constructs two float object.
2. User calls add method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 67:
1. User constructs two double object.
2. User calls add method.
3. User calls mul method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 68:
1. User constructs two integer object.
2. User calls add method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 69:
1. User constructs two float object.
2. User calls add method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 70:
1. User constructs two double object.
2. User calls add method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 71:
1. User constructs two integer object.
2. User calls sub method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 72:
1. User constructs two float object.
2. User calls sub method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 73:
1. User constructs two double object.
2. User calls sub method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 74:
1. User constructs two integer object.
2. User calls mul method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 75:
1. User constructs two float object.
2. User calls mul method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 76:
1. User constructs two double object.
2. User calls mul method.
3. User calls div method.
4. User requests the processed values via get() method.
5. User exits application.

Normal Scenario 77:
1. User constructs two integer object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 78:
1. User constructs two float object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 79:
1. User constructs two double object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 80:
1. User constructs two integer object.
2. User calls sub method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 81:
1. User constructs two float object.
2. User calls sub method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 82:
1. User constructs two double object.
2. User calls sub method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 83:
1. User constructs two integer object.
2. User calls add method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 84:
1. User constructs two float object.
2. User calls add method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 85:
1. User constructs two double object.
2. User calls add method.
3. User calls mul method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 86:
1. User constructs two integer object.
2. User calls add method.
3. User calls sub method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 87:
1. User constructs two float object.
2. User calls add method.
3. User calls sub method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 88:
1. User constructs two double object.
2. User calls add method.
3. User calls sub method.
4. User calls div method.
5. User requests the processed values via get() method.
6. User exits application.

Normal Scenario 89:
1. User constructs two integer object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User calls div method.
6. User requests the processed values via get() method.
7. User exits application.

Normal Scenario 90:
1. User constructs two float object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User calls div method.
6. User requests the processed values via get() method.
7. User exits application.

Normal Scenario 91:
1. User constructs two double object.
2. User calls add method.
3. User calls sub method.
4. User calls mul method.
5. User calls div method.
6. User requests the processed values via get() method.
7. User exits application.

JAVA SOURCE CODE

```
/**
      @author       Marco Martinez
      @fileName     Number.java
      @version      1.0
      @description  This program will create and manipulate Number objects.
      @date         12/8/2018

         Program Change Log
         ==================
         Name        Date     Description
         Marco       12/8     Create baseline for Number.java
 */

public class Number
{
   // INSTANCE VARIABLE DECLARATION
   private int integerPart;
   private double realPart;

   // CLASS CONSTRUCTORS
   public Number(){}

   public Number(int integer)
   {
      this.integerPart = integer;
      this.realPart = 0.00;
```

```
}

public Number(double real)
{
   this.integerPart = (int) real;
   this.realPart = real % 1;
}

public Number(int integer,double real)
{
   this.integerPart = integer;
   this.integerPart += (int) real;
   this.realPart = real % 1;
}

public Number(float real)
{
   this.integerPart = (int) real;
   this.realPart = (double) real % 1;
}

public Number(int integer,float real)
{
   this.integerPart = integer;
   this.integerPart = (int) real;
   this.realPart = real % 1;
}

public Number(Number number)
{
   this.integerPart = number.getInt();
   this.realPart = number.getReal();
}

// CHANGE STATE SERVICES
public void setInt(int integer)
{
   this.integerPart = integer;
}

public void setInt(double integer)
{
   this.integerPart = (int) integer;
}

public void setInt(float integer)
{
   this.integerPart = (int) integer;
}

public void setReal(int real)
{
   this.realPart = (double) real % 1;
}
```

```
   public void setReal(double real)
   {
      this.realPart = real % 1;
   }

   public void setReal(float real)
   {
      this.realPart = (double) real % 1;
   }

   // READ STATE SERVICES
   public int getInt()
   {
      return this.integerPart;
   }

   public double getReal()
   {
      return this.realPart;
   }

/**
      @author       Marco Martinez
      @fileName     NumberMath.java
      @version      1.0
      @description  This program will create and manipulate NumberMath objects.
      @date         12/8/2018

         Program Change Log
         ==================
         Name        Date     Description
         Marco       12/8     Create baseline for NumberMath.java
 */

public class NumberMath
{
   // INSTANCE VARIABLE DECLARATION
   private Number a;
   private Number b;
   private Number result;

   // CLASS CONSTRUCTORS
   public NumberMath(){}

   public NumberMath(int integer)
   {
      this.a = new Number(integer);
      this.b = new Number(0);
      this.result = new Number(0);
   }

   public NumberMath(double real)
   {
      this.a = new Number(real);
      this.b = new Number(0);
```

```
      this.result = new Number(0);
   }

   public NumberMath(int integer,double real)
   {
      this.a = new Number(integer,real);
      this.b = new Number(0);
      this.result = new Number(0);
   }

   public NumberMath(float real)
   {
      this.a = new Number(real);
      this.b = new Number(0);
      this.result = new Number(0);
   }

   public NumberMath(int integer,float real)
   {
      this.a = new Number(integer,real);
      this.b = new Number(0);
      this.result = new Number(0);
   }

   public NumberMath(Number number)
   {
      this.a = new Number(number);
      this.b = new Number(0);
      this.result = new Number(0);
   }

   public NumberMath(int integerFirst,int integerSecond)
   {
      this.a = new Number(integerFirst);
      this.b = new Number(integerSecond);
      this.result = new Number(0);
   }

   public NumberMath(double realFirst,double realSecond)
   {
      this.a = new Number(realFirst);
      this.b = new Number(realSecond);
      this.result = new Number(0);
   }

   public NumberMath(int integerFirst,double realFirst,int integerSecond,double
realSecond)
   {
      this.a = new Number(integerFirst,realFirst);
      this.b = new Number(integerSecond,realSecond);
      this.result = new Number(0);
   }

   public NumberMath(float realFirst,float realSecond)
   {
```

```
   this.a = new Number(realFirst);
   this.b = new Number(realSecond);
   this.result = new Number(0);
}

public NumberMath(int integerFirst,float realFirst,int integerSecond,float realSecond)
{
   this.a = new Number(integerFirst,realFirst);
   this.b = new Number(integerSecond,realSecond);
   this.result = new Number(0);
}

public NumberMath(Number numberFirst,Number numberSecond)
{
   this.a = new Number(numberFirst);
   this.b = new Number(numberSecond);
   this.result = new Number(0);
}

// CHANGE STATE SERVICES
public void setA(int integer)
{
   this.a.setInt(integer);
   this.a.setReal(0.00);
}

public void setA(float real)
{
   this.a.setInt((int)real);
   this.a.setReal(real);
}

public void setA(double real)
{
   this.a.setInt((int)real);
   this.a.setReal(real);
}

public void setA(int integer,float real)
{
   this.a.setInt(integer);
   this.a.setReal(real);
}

public void setA(int integer,double real)
{
   this.a.setInt(integer);
   this.a.setReal(real);
}

public void setB(int integer)
{
   this.b.setInt(integer);
   this.b.setReal(0.00);
}
```

```
   public void setB(float real)
   {
      this.b.setInt((int)real);
      this.b.setReal(real);
   }

   public void setB(double real)
   {
      this.b.setInt((int)real);
      this.b.setReal(real);
   }

   public void setB(int integer,float real)
   {
      this.b.setInt(integer);
      this.b.setReal(real);
   }

   public void setB(int integer,double real)
   {
      this.b.setInt(integer);
      this.b.setReal(real);
   }

   public void add()
   {
      this.result.setInt((int)(this.a.getInt() + this.b.getInt()) + (this.a.getReal() +
this.b.getReal()));
      this.result.setReal((this.a.getInt() + this.b.getInt()) + (this.a.getReal() +
this.b.getReal()));
   }

   public void sub()
   {
      this.result.setInt((int)(this.a.getInt() - this.b.getInt()) + (this.a.getReal() -
this.b.getReal()));
      this.result.setReal((this.a.getInt() - this.b.getInt()) + (this.a.getReal() -
this.b.getReal()));
   }

   public void mul()
   {
      this.result.setInt((int)(this.a.getInt() * this.b.getInt()) + (this.a.getReal() *
this.b.getReal()));
      this.result.setReal((this.a.getInt() * this.b.getInt()) + (this.a.getReal() *
this.b.getReal()));
   }

   public void div()
   {
      double temp;
      temp = (this.a.getInt() / this.b.getInt()) + (this.a.getReal() / this.b.getReal());
      this.result.setInt((int)(this.a.getInt() / this.b.getInt()) + (this.a.getReal() /
this.b.getReal()));
```

```
      this.result.setReal((this.a.getInt() / this.b.getInt()) + (this.a.getReal() /
this.b.getReal()));
   }
   // READ STATE SERVICES
   public Number getA()
   {
      return this.a;
   }

   public Number getB()
   {
      return this.b;
   }

   public Number getResult()
   {
      return this.result;
   }

   public String toString()
   {
      return "a = " + this.a.toString() + ", b = " + this.b.toString() + ", result = " +
this.result.toString();
   }
}

/**
      @author       Marco Martinez
      @fileName     AppDriver.java
      @version      1.0
      @description  This program will test Number/NumberMath objects.
      @date         12/8/2018

         Program Change Log
         ==================
         Name        Date     Description
         Marco       12/8     Create baseline for AppDriver.java
 */

public class AppDriver
{
   public static void main(String[] args)
   {
      NumberMath myNum1 = new NumberMath();
      NumberMath myNum2 = new NumberMath((int)2);
      NumberMath myNum3 = new NumberMath((int)3,(int)4);
      NumberMath myNum4 = new NumberMath((int)5,5.5,(int)3,9.4);
      NumberMath myNum5 = new NumberMath((int)5,(float).44,(int)9,(float).333);
      NumberMath myNum6 = new NumberMath(myNum5.getA(),myNum5.getB());
      NumberMath myNum = new NumberMath(10.5,2.25);
      myNum.add();
      System.out.println("The result of adding "+ (myNum.getA()).toString() +" and "+
(myNum.getB()).toString() +": "+myNum.toString());
      System.out.println();
      myNum.sub();
```

```
      System.out.println("The result of subtracting "+ (myNum.getA()).toString() +" and
"+ (myNum.getB()).toString() +": "+myNum.toString());
      System.out.println();
      myNum.mul();
      System.out.println("The result of multiplying "+ (myNum.getA()).toString() +" and
"+ (myNum.getB()).toString() +": "+myNum.toString());
      System.out.println();
      myNum.div();
      System.out.println("The result of dividing "+ (myNum.getA()).toString() +" and "+
(myNum.getB()).toString() +": "+myNum.toString());
   }
}
```

SCREENSHOT