CS 2420
Fall semester section 01
Ricardo Gonzalez

Review: Chapter 3

Written question:

3.1: information hiding is used in object oriented used to restrict accesses to some objects. Encapsulation is bundled data and methods into one unit to be used together. Java supports these concepts by using getters and setters you can access the restricted objects, and you use classes to make those units.

3.2: public section of the class mean it can be accessed by anyone, while private of the class can only by accessed by members of the same class can only accesses.

3.3: the purpose of the constructor is the initializes for newly created objects to be used.

3.4: if you don't provide a constructor then there will be a no argument constructor will be used.

3.6: if the void is included in the constructer it will act as a method and not the constructer.

3.10: difference between static and instance field, static have to use static keyword instance don't have keywords, can invoke static field by the class name instance need an object to invoke it, if you use a static it can be used in many class or methods and only has to be created the one time, while can only use instance has to have copies if used elsewhere.

In Theory:

3.14: if you create a private constructor, then only the one object can be created and other classes do not have permission to make an object from that class because it is private.

3.15: a. yes it would work. b. yes It would still work if it was uncommented.

Programming:

3.18: here is my combination class that has the combination private so it can be accessed then I have a boolean isLockedopened method to check if the lock is open so people can change the combination. Then I have my method to check if the inputs are correct to have the lock opened and if is opened then I have a changeCombo method that if they got the right combo then you can put in a new combo

```
import java.util.Scanner;

class combination {
    private int x = 3;
    private int y = 8;
    private int a = 1;

    boolean lockopened = false;


    public boolean isLockopened() {
        return lockopened;
    }

    public void setLockopened(boolean lockopened) {
        this.lockopened = lockopened;
    }

    public combination(int x, int y, int a) {
        this.x = x;
        this.y = y;
        this.a = a;
    }

    public void Opened(int x, int y, int a) {
        if(x == 3 && y == 8 && a == 1) {
            System.out.println("correct! you opend the lock.");
            lockopened = true;
        }
        else {
            System.out.println("wrong combination");
        }
    }

    public void changeCombo() {
        Scanner sc = new Scanner(System.in);
        System.out.println("what is the new 3 combination you would like? ");
        x = sc.nextInt();
        y = sc.nextInt();
        a = sc.nextInt();
        System.out.println("new code is: " + x + " " + y + " " + a);
    }
}
```

then I have the main which executes everything out.

```java
import java.util.Scanner;
public class chapter3programing  {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int input1 = 0;
        int input2 = 0;
        int input3 = 0;


        System.out.println("guess the the 3 digit combination :");
        input1 = sc.nextInt();
        input2 = sc.nextInt();
        input3 = sc.nextInt();
        combination lock = new combination(input1, input2, input3);

        lock.Opened(input1, input2, input3);

        if(lock.isLockopened() == true) {
            lock.changeCombo();
        }
    }
```

what you get if you enter the wrong combination.

```
guess the the 3 digit combination :
4
6
8
wrong combination
```

what you get if you enter the right combination.

```
guess the the 3 digit combination :
3
8
1
correct! you opend the lock.
what is the new 3 combination you would like?
3
2
1
new code is: 3 2 1
```

3.31:

```java
import java.math.BigDecimal;

public class BigRational
{
public static final BigRational ZERO = new BigRational( );
    private BigDecimal input1;
    private BigDecimal input2;
    private String str;

   public BigRational( ){
           this(BigDecimal.ZERO);
       }
    public BigRational( BigDecimal input1 ){
           this(input1, BigDecimal.ZERO);
        }
    public BigRational( BigDecimal n, BigDecimal d ){
        input1 = n;
        input2 = d;
    }
    public BigRational( String str ){
    if( str.length( ) == 0 )
        throw new IllegalArgumentException( "Zero-length string" );

        // Check for '+' and '-'
        int slashIndex1 = str.indexOf('+');
        int slashIndex2 = str.indexOf( '-' );
        if(slashIndex1 == -1 && slashIndex2 == -1)
        {
            input1 = new BigDecimal( str.trim( ) );
            input2 = BigDecimal.ZERO;
        }
        else if(slashIndex2 == -1)
        {
            input1 = new BigDecimal( str.substring( 0, slashIndex1 ).trim( ) );
            input2 = new BigDecimal( str.substring( slashIndex1 + 1 ).trim( ).replace("i", "") );
        }
        else if(slashIndex1 == -1) {
            input1 = new BigDecimal( str.substring( 0, slashIndex2 ).trim( ) );
            input2 = new BigDecimal( str.substring( slashIndex2 + 1 ).trim( ).replace("i", ""));
        }
    }

     public BigDecimal getInput1() {
        return input1;
    }
    public void setInput1(BigDecimal input1) {
        this.input1 = input1;
    }
    public BigDecimal getInput2() {
        return input2;
    }
    public void setInput2(BigDecimal input2) {
        this.input2 = input2;
    }

    public BigRational add( BigRational other ){
        BigDecimal n = getInput1().add(other.getInput1());
        BigDecimal d = getInput2().add(other.getInput2());
        System.out.println(new BigRational(n, d));
        return new BigRational(n, d);
     }
     public BigRational subtract( BigRational other ){
         BigDecimal n = getInput1().subtract(other.getInput1());
         BigDecimal d = getInput2().subtract(other.getInput2());
         System.out.println(new BigRational(n, d));
         return new BigRational(n, d);
     }
```

```java
    public boolean equals( Object other )
    {
        if( ! ( other instanceof BigRational ) )
            return false;

        BigRational rhs = (BigRational) other;

        return getInput1().equals( rhs.getInput1() ) && getInput2().equals( rhs.getInput2() );
    }
    public String toString()
    {
        return getInput1().toString() + " + " + getInput2().toString() + "i";
    }

}
```

Everything I did in bigraiton class

And everything I had in my main to test out

```java
import java.math.BigDecimal;

public class Main {

    public static void main(String[] args) {
        BigDecimal inpt1 = new BigDecimal("2.4");
        BigDecimal inpt2 = new BigDecimal("3.1");
        BigDecimal inpt3 = new BigDecimal("4.3");

        BigRational b1 = new BigRational();
        BigRational b2 = new BigRational(inpt1);
        BigRational b5 = new BigRational(inpt2, inpt2);
        BigRational b3 = new BigRational(inpt1, inpt3);
        BigRational b4 = new BigRational("1.23+4i");

        System.out.println(b1.toString());

        System.out.println(b2.toString());
        System.out.println(b1.toString());
        b2.add(b1);
        b2.equals(b1);

        System.out.println(b3.toString());
        System.out.println(b2.toString());
        b3.add(b2);
        b3.equals(b3);

        System.out.println(b2.toString());
        System.out.println(b1.toString());
        b2.subtract(b1);
        b2.equals(b1);

        System.out.println(b5.toString());
        System.out.println(b3.toString());
        b5.subtract(b3);
        b3.equals(b3);

        System.out.println(b4);

    }
}
```

```
0 + 0i
2.4 + 0i
0 + 0i
2.4 + 0i
2.4 + 4.3i
2.4 + 0i
4.8 + 4.3i
2.4 + 0i
0 + 0i
2.4 + 0i
3.1 + 3.1i
2.4 + 4.3i
0.7 + -1.2i
1.23 + 4i
```

the answer I got from testing my main all make sense and the add and subtract method are both working as well as the toStirng method and the equals method.