Gregory Prosper COP 4331 HW3

Question 3.1

a. Encapsulation makes sure that changes to one part of software will not have a large side effect on other parts.
b. When there is not a verifiable precondition such as checking if a file exist, which could be deleted by another program immediately after it is checked.
c. It is more efficient for a method to produce the same results no mater how many times it is called. Side effects would make methods less predictable.

Question 3.2

Complex.java

```
package math;

/**
 * Class for representing complex numbers
 */
public class Complex {

	/**
	 * Constructor that takes real and imaginary numbers
	 *
	 * @param real
	 *            real number
	 * @param imaginary
	 *            imaginary number
	 */
	Complex(double real, double imaginary) {
		this.real = real;
		this.imaginary = imaginary;
	}

	/**
	 * Constructor that takes real and sets imaginary number to 0
	 *
	 * @param real
	 *            real number
	 */
	Complex(double real) {
		this.real = real;
		this.imaginary = 0;
```

```
}

/**
 * Returns complex number as a string
 *
 * @return Complex number
 */
public String toString() {
	String s;
	if (this.imaginary > 0) {
		s = String.format("%.2f + %.2fi", this.real, this.imaginary);
	} else if (this.imaginary == 0) {
		s = String.format("%.2f", this.real);
	} else {
		s = String.format("%.2f - %.2fi", this.real, this.imaginary * -1);
	}

	return s;
}

/**
 * Returns the complex number's real number value
 *
 * @return Complex number's real number
 */
public double r() {
	return this.real;
}

/**
 * Returns the complex number's imaginary number value
 *
 * @return Complex number's imaginary number
 */
public double i() {
	return this.imaginary;
}

/**
 * Returns the sum between this complex number and another
 *
 * @return Complex number that is the sum between this complex number and
 *         another
 */
public Complex add(Complex other) {
	double realTemp = this.real + other.real;
	double imaginaryTemp = this.imaginary + other.imaginary;
	return new Complex(realTemp, imaginaryTemp);
}

/**
 * Returns the difference between this complex number and another
 *
 * @return Complex number that is the difference between this complex number
 *         and another
 */
public Complex sub(Complex other) {
	double realTemp = this.real - other.real;
	double imaginaryTemp = this.imaginary - other.imaginary;
	return new Complex(realTemp, imaginaryTemp);
```

```
}

/**
 * Returns the Conjugate of this complex number
 *
 * @return Conjugate of this complex number
 */
public Complex conj() {
	double imaginaryTemp = this.imaginary * -1;
	return new Complex(this.real, imaginaryTemp);
}

/**
 * Returns the multiplication between this complex number and another
 *
 * @return Complex number that is the multiplication between this complex
 *         number and another
 */
public Complex mult(Complex other) {
	double first = this.real * other.real;
	double outerImaginary = this.real * other.imaginary;
	double innerImaginary = this.imaginary * other.real;
	double last = this.imaginary * other.imaginary * -1;

	return new Complex(first + last, outerImaginary + innerImaginary);
}

/**
 * Returns the division between this complex number and another
 *
 * @return Complex number that is the division between this complex number
 *         and another
 * @throws IllegalArgumentException
 *             if d.real == 0
 */
public Complex div(Complex other) {
	Complex conjugate = other.conj();
	Complex n = new Complex(this.real, this.imaginary).mult(conjugate);
	Complex d = new Complex(other.real, other.imaginary).mult(conjugate);
	if (d.real == 0) {
		throw new IllegalArgumentException("Argument 'divisor' is 0");
	}
	return new Complex(n.real / d.real, n.imaginary / d.real);

}

/**
 * Test to see if this complex number is equal to another
 *
 * @return boolean that is true if this complex number is equal to another
 */
public boolean equals(Complex other) {
	if (Math.abs(this.real - other.real) < Complex.MARGIN
		&& Math.abs(this.imaginary - other.imaginary) < Complex.MARGIN) {
		return true;
	} else
		return false;
}

private final double real;
```

```
	private final double imaginary;
	private static final double MARGIN = 0.000000001;
}
```

ComplexTester.java

```
package math;

public class ComplexTester {

	/**
	 * @param args
	 */
	public static void main(String[] args) {

		Complex n1 = new Complex(5, 2);
		Complex n2 = new Complex(7, 4);
		Complex n3 = new Complex(5, 2);

		// addition
		Complex N1plusN2 = n1.add(n2);

		// subtraction
		Complex N1subN2 = n1.sub(n2);

		// multiplication
		Complex N1multN2 = n1.mult(n2);

		// division
		Complex N1divN2 = n1.div(n2);

		// conjugate
		n1 = new Complex(5, 2);
		Complex N1conjugate = n1.conj();

		// equal
		boolean N1equalN3 = n1.equals(n3);

		// equal
		boolean N1equalN2 = n1.equals(n2);

		System.out.print("n1 real number: ");
		System.out.println(n1.r());
		System.out.print("n1 imaginary number: ");
		System.out.println(n1.i());
		System.out.print("n2 real number: ");
		System.out.println(n2.r());
		System.out.print("n2 imaginary number: ");
		System.out.println(n2.i());
		System.out.printf("Addition of (%s) and (%s):\n", n1.toString(),
n2.toString());
		System.out.println(N1plusN2.toString());
		System.out.printf("Subtraction of (%s) and (%s):\n", n1.toString(),
n2.toString());
		System.out.println(N1subN2.toString());
		System.out.printf("Multiplication of (%s) and (%s):\n", n1.toString(),
n2.toString());
```

```
                System.out.println(N1multN2.toString());
                System.out.printf("Division of (%s) and (%s):\n", n1.toString(),
n2.toString());
                System.out.println(N1divN2.toString());
                System.out.println("Conugate of n1:");
                System.out.println(N1conjugate.toString());
                System.out.println("Equality of n1 and n3:");
                System.out.println(N1equalN3);
                System.out.println("Equality of n1 and n2:");
                System.out.println(N1equalN2);

        }

}
```

ComplexTest.java (JUnit Test)

```
package math;

import org.junit.* ;
import static org.junit.Assert.* ;

public class ComplexTest {

        @Test
        public void testEqualsComplex() {
                System.out.println("run test equals()");
                double a = 1, b = 2;
                Complex x = new Complex(a, b);
                Complex y = new Complex(a, b);
                assertTrue(x.equals(y));
        }

        @Test
        public void testToString() {
                System.out.println("run test toString()");
                double a = 1, b = 2;
                Complex x = new Complex(a, b);
                assertEquals("1.00 + 2.00i", x.toString());
        }

        @Test
        public void testR() {
                System.out.println("run test r()");
                double a = 1, b = 2;
                Complex x = new Complex(a, b);
                assert a == x.r();
        }

        @Test
        public void testI() {
                System.out.println("run test i()");
                double a = 1, b = 2;
                Complex x = new Complex(a, b);
                assert b == x.i();
        }

        @Test
        public void testAdd() {
```

```
        System.out.println("run test add()");
        double a = 1, b = 2, c = -3, d = 4;
        double e = a + c, f = b + d;
        Complex x = new Complex(a, b);
        Complex y = new Complex(c, d);

        Complex w = x.add(y);

        Complex z = new Complex(e, f);

        // set up Complex objects
        // test condition using the Complex equals() method:
        assertTrue(z.equals(w));
}

@Test
public void testSub() {
        System.out.println("run test sub()");
        double a = 1, b = 2, c = -3, d = 4;
        double e = a - c, f = b - d;
        Complex x = new Complex(a, b);
        Complex y = new Complex(c, d);

        Complex w = x.sub(y);

        Complex z = new Complex(e, f);

        // set up Complex objects
        // test condition using the Complex equals() method:
        assertTrue(z.equals(w));
}

@Test
public void testConj() {
        System.out.println("run test conj()");
        double a = 1, b = 2;
        Complex x = new Complex(a, b);

        assertTrue(x.conj().equals(new Complex(1, -2)));
}

@Test
public void testMult() {
        System.out.println("run test sub()");
        Complex x = new Complex(2, 3);
        Complex y = new Complex(4, 5);

        Complex w = x.mult(y);

        Complex z = new Complex(-7,22);

        // set up Complex objects
        // test condition using the Complex equals() method:
        assertTrue(z.equals(w));
}

@Test
public void testDiv() {
        System.out.println("run test div()");
        Complex x = new Complex(4, 2);
```

```
                Complex y = new Complex(3, -1);
                Complex z = x.div(y);
                // set up Complex objects
                // test condition using the Complex equals() method:
                assertTrue(z.equals(new Complex(1,1)));
        }

}
```

Question 4.1

Student.java

```
package student;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Date;

/**
 * Class for representing Students
 */
public class Student {

        /**
         * Constructor which takes name and date object
         */
        Student(String name, Date whenEnrolled) {
                this.name = name;
                this.enrollment = (Date) whenEnrolled.clone();
        }

        /**
         * Returns student name
         *
         * @return student name as string
         */
        public String getName() {
                return this.name;
        }


        /**
         * Returns student name
         *
         * @return student enrollment date as Date object
         */
        public Date getEnrollment() {
                return (Date) this.enrollment.clone();
        }

        /**
         * Returns Comparator to sort by name
         *
         * @return Comparator to sort by name
```

```
 */
public static Comparator<Student> getCompByName() {
	return new Comparator<Student>() {
		public int compare(Student student1, Student student2) {
			return student1.getName().compareTo(student2.getName());
		}
	};
}

/**
 * Returns Comparator to sort by enrollment date
 *
 * @return Comparator to sort by enrollment date
 */
public static Comparator<Student> getCompByDate() {
	return new Comparator<Student>() {
		public int compare(Student student1, Student student2) {
			return student1.getEnrollment().compareTo(
					student2.getEnrollment());
		}
	};
}

public static void main(String[] args) {
	ArrayList<Student> studentList = new ArrayList<>();
	Date date = new Date();

	date.setDate(31);
	date.setMonth(1);
	date.setYear(1992);
	Student s1 = new Student("Prosper, Gregory", date);
	studentList.add(s1);

	date.setDate(30);
	date.setMonth(7);
	date.setYear(1992);
	Student s2 = new Student("Jean, Mideline", date);
	studentList.add(s2);

	date.setDate(18);
	date.setMonth(8);
	date.setYear(2010);
	Student s3 = new Student("Prosper, Kenny", date);
	studentList.add(s3);

	date.setDate(11);
	date.setMonth(6);
	date.setYear(2001);
	Student s4 = new Student("Miller, Jerry", date);
	studentList.add(s4);

	System.out.println("No Sort:");
	for (Student student : studentList) {
		System.out.println(student.getName());
	}

	System.out.println("\nSort By Name:");
	Collections.sort(studentList, Student.getCompByName());

	for (Student student : studentList) {
```

```
                        System.out.println(student.getName());
                }

                System.out.println("\nSort By Date:");
                Collections.sort(studentList, Student.getCompByDate());

                for (Student student : studentList) {
                        System.out.println(student.getName());
                }

        }

        private String name;
        private Date enrollment;
}
```

Question 4.2

Gui.java

```
package gui;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.*;

public class Gui {

        private static JButton createButton(int i) {
                String[] c = { "Green", "Blue", "Red" };
                JButton button = new JButton(c[i]);
                button.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent action) {
                                System.out.println(action.getActionCommand());
                                if (action.getActionCommand() == "Green") {
                                        icon.setColor(Color.GREEN);
                                        label.repaint();
                                } else if (action.getActionCommand() == "Red") {
                                        icon.setColor(Color.RED);
                                        label.repaint();
                                } else if (action.getActionCommand() == "Blue") {
                                        icon.setColor(Color.BLUE);
                                        label.repaint();
                                }
                        }
                }

                );
                return button;
        }


        public static void main(String[] args) {
```

```
                JFrame frame = new JFrame("Color Picker");
                ArrayList<JButton> buttons = new ArrayList<>();

                icon = new ColorIcon(50);
                label = new JLabel(icon);

                for (int i = 0; i < 3; i++) {
                        buttons.add(createButton(i));
                }

                frame.setLayout(new FlowLayout(FlowLayout.CENTER));
                frame.add(label);
                frame.add(buttons.get(0));
                frame.add(buttons.get(1));
                frame.add(buttons.get(2));

                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.pack();
                frame.setVisible(true);

        }

        private static ColorIcon icon;
        private static JLabel label;
}
```

ColorIcon.java

```
package gui;

import java.awt.*;
import java.awt.geom.*;

import javax.swing.Icon;

public class ColorIcon implements Icon {

        public ColorIcon(int aSize, Color c) {
                this.size = aSize;
                this.color = c;
        }

        public ColorIcon(int aSize) {
                size = aSize;
                this.color = Color.RED;
        }

        public int getIconWidth() {
                return size;
        }

        public int getIconHeight() {
                return size;
        }

        public void setColor(Color c) {
                this.color = c;
        }
```

```
        public void paintIcon(Component c, Graphics g, int x, int y) {
                Graphics2D g2 = (Graphics2D) g;
                Ellipse2D.Double shape = new Ellipse2D.Double(x, y, size, size);
                g2.setColor(this.color);
                g2.fill(shape);
        }

        private int size;
        private Color color;

}
```