Name:	Dan Cassidy
Class:	CSCI-C 490, Mobile Application Development
Assignment:	Homework 2 Part 1
Date:	2015-07-14

```java
/*-----------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy
 * Date:        2015-07-14
 * Assignment:  HW2-1
 * Source File: Rational.java
 * Language:    Java
 * Course:      CSCI-C 490, Android Programming, MoWe 08:00
 -----------------------------------------------------------------------------------------------*/

/**
 * Small class that represents rational numbers in accordance with homework assignment 2-1. By
 * default, the numerator is set to 0 and the denominator is set to 1.
 *
 * @author Dan Cassidy
 */
public class Rational
{
    private int numerator = 0;
    private int denominator = 1;

    /**
     * Default constructor.
     */
    public Rational()
    {
        // Do nothing, numerator and denominator are already set to 0 and 1 respectively.
    }

    /**
     * 1-parameter constructor. Takes a whole number and creates a new Rational object based upon
     * that. This means that the denominator will be 1.
     *
     * @param wholeNumber The number that will be stored in the numerator.
     */
    public Rational(int wholeNumber)
    {
        this.setNumerator(wholeNumber);
        // Don't have to set denominator because it's 1 by default.
    }

    /**
     * 2-parameter constructor. Takes two arguments that will be the numerator and denominator of
     * the resultant Rational object.
     *
     * @param numerator The number that will be the numerator.
     * @param denominator The number that will be the denominator.
     */
    public Rational(int numerator, int denominator)
    {
        this.setNumerator(numerator);
        this.setDenominator(denominator);
    }

    // BEGIN GETTERS AND SETTERS -->
    public int getDenominator()
    {
        return this.denominator;
    }

    /**
```

```
     * @param denominator The number to set the denominator to.
     * @throws IllegalArgumentException if <b>denominator</b> is 0.
     */
    public void setDenominator(int denominator)
    {
        if (denominator == 0)
            throw new IllegalArgumentException("Cannot have a denominator of 0.");

        this.denominator = denominator;
        this.simplify();
    }

    public int getNumerator()
    {
        return this.numerator;
    }

    /**
     * @param numerator The number to set the numerator to.
     */
    public void setNumerator(int numerator)
    {
        this.numerator = numerator;
        this.simplify();
    }
    // <-- END GETTERS AND SETTERS

    /**
     * Addition method that takes two Rational objects and adds them together.
     *
     * @param r1 The first of two Rational objects that will be added together.
     * @param r2 The second of two Rational objects that will be added together.
     * @return Rational object, containing the result of the operation.
     * @throws NullPointerException if <b>r1</b> or <b>r2</b> is null.
     */
    public static Rational add(Rational r1, Rational r2)
    {
        if (r1 == null || r2 == null)
            throw new NullPointerException();

        if (r1.denominator == r2.denominator)
            return new Rational(r1.numerator + r2.numerator, r1.denominator);
        else
            return new Rational(r1.numerator * r2.denominator + r2.numerator * r1.denominator,
                    r1.denominator * r2.denominator);
    }

    /**
     * Subtraction method that takes two Rational objects and subtracts the second from the first.
     *
     * @param r1 The first of two Rational objects. Will be subtracted from by the second.
     * @param r2 The second of two Rational objects. Will subtract from the first.
     * @return Rational object, containing the result of the operation.
     * @throws NullPointerException if <b>r1</b> or <b>r2</b> is null.
     */
    public static Rational subtract(Rational r1, Rational r2)
    {
        if (r1 == null || r2 == null)
            throw new NullPointerException();

```

```java
        if (r1.denominator == r2.denominator)
            return new Rational(r1.numerator - r2.numerator, r1.denominator);
        else
            return new Rational(r1.numerator * r2.denominator - r2.numerator * r1.denominator,
                    r1.denominator * r2.denominator);
    }

    /**
     * Multiplication method that takes two Rational objects and multiplies them together.
     *
     * @param r1 The first of two Rational objects that will be multiplied together.
     * @param r2 The second of two Rational objects that will be multiplied together.
     * @return Rational object, containing the result of the operation.
     * @throws NullPointerException if <b>r1</b> or <b>r2</b> is null.
     */
    public static Rational multiply(Rational r1, Rational r2)
    {
        if (r1 == null || r2 == null)
            throw new NullPointerException();

        return new Rational(r1.numerator * r2.numerator, r1.denominator * r2.denominator);
    }

    /**
     * Division method that takes two Rational objects and divides the first by the second.
     *
     * @param r1 The first of two Rational objects. Will be divided by the second.
     * @param r2 The second of two Rational objects. Will divide the first.
     * @return Rational object, containing the result of the operation.
     * @throws NullPointerException if <b>r1</b> or <b>r2</b> is null.
     */
    public static Rational divide(Rational r1, Rational r2)
    {
        if (r1 == null || r2 == null)
            throw new NullPointerException();

        return new Rational(r1.numerator * r2.denominator, r1.denominator * r2.numerator);
    }

    /**
     * Equals method that compares this object to another Rational object.
     * @param other A Rational object that will be compared against to determine equality.
     * @return boolean, representing whether this object is equal to the other object or not.
     */
    public boolean equals(Rational other)
    {
        if (other == null)
            return false;

        return (this.numerator * other.denominator == other.numerator * this.denominator);
    }

    /**
     * Override of the toString() method. Returns a String representation of the object.
     * @return String containing either the numerator and denominator separated by a forward slash
     * if denominator is not equal to 1, or just the numerator if the denominator is equal to 1.
     */
    public String toString()
    {
        if (this.denominator == 1)
```

```
            return "" + this.numerator;

        return this.numerator + "/" + this.denominator;
    }

    /**
     * Simplifies the current Rational object (this) by putting any negative signs in the numerator
     * and dividing both the numerator and the denominator by their greatest common divisor.
     */
    private void simplify()
    {
        if (this.denominator < 0)
        {
            this.numerator *= -1;
            this.denominator *= -1;
        }

        if (this.denominator != 1)
        {
            int gcd = gcd(this.numerator < 0 ? this.numerator * -1 : this.numerator,
                    this.denominator);
            this.numerator /= gcd;
            this.denominator /= gcd;
        }
    }

    /**
     * Iterative binary greatest common divisor (GCD) algorithm (Stein's algorithm) from
     * <a href="https://en.wikipedia.org/wiki/Binary_GCD_algorithm">Wikipedia</a>.
     *
     * @param u The first of two non-negative numbers to find the GCD of.
     * @param v The second of two non-negative numbers to find the GCD of.
     * @return An integer representing the greatest common divisor of <b>u</b> and <b>v</b>.
     * @throws IllegalArgumentException if either <b>u</b> or <b>v</b> are negative.
     */
    private int gcd(int u, int v)
    {
        if (u < 0 || v < 0)
            throw new IllegalArgumentException("Arguments must be non-negative.");

        // If either argument is 0, return the other argument.
        if (u == 0)
            return v;
        if (v == 0)
            return u;

        // Find the greatest power of 2 dividing both u and v.
        int shift;
        for (shift = 0; ((u | v) & 1) == 0; ++shift)
        {
            u >>= 1;
            v >>= 1;
        }

        // Make u odd.
        while ((u & 1) == 0)
            u >>= 1;

        do
        {
```

```
            // Remove all factors of 2 in v.
            while ((v & 1) == 0)
                v >>= 1;

            // Now u and v are both odd. Swap if necessary so u <= v, then set v = v - u.
            if (u > v)
            {
                int t = v;
                v = u;
                u = t;
            }
            v = v - u;
        } while (v != 0);

        // Restore common factors of 2.
        return u << shift;
    }
}
```

Testing the Rational class via RationalTest.

```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-1\bin>java
RationalTest
Number is: 0
Number is: 5
Number is: 1/4
Rational1's numerator is: 0
Rational1's denominator is: 1
Changing rational1's numerator.
Rational1 is : 3
Chaning rational1's denominator.
Rational1 is: 3/4
rational1 is equal to rational2: false
Rational is equal to a copy of rational2: true
Adding 1/4 and 3/4: 1
Adding 3/4 and 5/1: 23/4
Subtracting 3/4 - 1/4: 1/2
Subtracting 5/1 - 3/4: 17/4
Multiplying 1/4 * 3/4: 3/16
Dividing 5/1 by 1/2: 10

C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-1\bin>_
```

Name:	Dan Cassidy
Class:	CSCI-C 490, Mobile Application Development
Assignment:	Homework 2 Part 2
Date:	2015-07-09

```java
/**
* Team copy constructor;  Performs a deep copy from the Competition class
* instead of just copying a reference to the same Competition objects.
* @param otherTeam The other team to deep-copy.
* @throws NullPointerException if <b>otherTeam</b> is null.
*/
public Team(Team otherTeam)
{
    // Verify that the method is not being passed a null.
    if (otherTeam == null)
        throw new NullPointerException();

    // Copy the basic fields from the other Team object.
    this.teamName = otherTeam.getTeamName();
    this.name1 = otherTeam.getName1();
    this.name2 = otherTeam.getName2();
    this.name3 = otherTeam.getName3();
    this.name4 = otherTeam.getName4();

    // Deep copy the Competition objects.
    Competition tempComp = otherTeam.getCompetition1();
    this.competition1 = new Competition(
            tempComp.getCompetitionName(),
            tempComp.getWinner(),
            tempComp.getRunnerup(),
            tempComp.getYear());
    tempComp = otherTeam.getCompetition2();
    this.competition2 = new Competition(
            tempComp.getCompetitionName(),
            tempComp.getWinner(),
            tempComp.getRunnerup(),
            tempComp.getYear());
}

```

Testing the deep copy constructor of the Team class.

```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-2\bin>java
Team
Team name :The Java Studs
Team member 1: Bill
Team member 2: Ted
Team member 3: Carol
Team member 4: Alice
Competitions:
Competition Name:ACM Programming Contest
Winner: The Java Studs
Runner-Up: Team University Java
Year: 2008
Competition Name:Antarctic Programming Contest
Winner: The Java Studs
Runner-Up: Frigid South
Year: 2010

Team name :Hellow World
Team member 1: Bill
Team member 2: Ted
Team member 3: Carol
Team member 4: Alice
Competitions:
Competition Name:ACM Programming Contest
Winner: Mmm Java
Runner-Up: Java By Night Java By Day
Year: 2011
Competition Name:Antarctic Programming Contest
Winner: The Java Studs
Runner-Up: Frigid South
Year: 2012


C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-2\bin>_
```

Name:	Dan Cassidy
Class:	CSCI-C 490, Mobile Application Development
Assignment:	Homework 2 Part Project
Date:	2015-07-09

```java
/*-----------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy
 * Date:        2015-07-09
 * Assignment:  HW2-Project
 * Source File: Administrator.java
 * Language:    Java
 * Course:      CSCI-C 490, Android Programming, MoWe 08:00
-----------------------------------------------------------------------------------------------*/
import java.util.Scanner;

/**
 * Implements the Administrator class as per the instructions for Homework 2-Project.<br>
 * Class Invariant: All objects have a name string, hire date, non-negative salary, title string,
 * area of responsibility string, and supervisor name string. A name string of "No name" indicates
 * no real name specified yet. A hire date of Jan 1, 1000 indicates no real hire date specified yet.
 *  A title of "No Title" indicates no real title specified yet. An area of "No Area" indicates no
 * real area of responsibility specified yet. A supervisor's name of "No Supervisor" indicates no
 * real supervisor specified yet.
 * @author Dan Cassidy
 */
public class Administrator extends SalariedEmployee
{
    private String title = "No Title";
    private String area = "No Area";
    private String supervisorsName = "No Supervisor";

    /**
     * Default constructor for an Administrator object.
     */
    public Administrator()
    {
        super();
        // Nothing else to do, defaults are set already.
    }

    /**
     * 6-parameter constructor for an Administrator object.
     * @param theName Employee's name.
     * @param theDate Employee's hire date.
     * @param theSalary Employee's yearly salary.
     * @param title Employee's title.
     * @param area Employee's area of responsibility.
     * @param supervisorsName Name of employee's supervisor.
     */
    public Administrator(String theName, Date theDate, double theSalary, String title,
            String area, String supervisorsName)
    {
        super(theName, theDate, theSalary);
        this.setTitle(title);
        this.setArea(area);
        this.setSupervisorsName(supervisorsName);
    }

    /**
     * Copy constructor.
     * @param originalObject Original Administrator object to duplicate.
     */
    public Administrator(Administrator originalObject)
    {
        super(originalObject);
```

```java
        this.setTitle(originalObject.getTitle());
        this.setArea(originalObject.getArea());
        this.setSupervisorsName(originalObject.getSupervisorsName());
    }

    // BEGIN GETTERS AND SETTERS -->
    public String getArea()
    {
        return this.area;
    }

    public void setArea(String area)
    {
        if (area == null)
            throw new NullPointerException("Area of Responsibility cannot be null.");
        else if (area.equals(""))
            throw new IllegalArgumentException("Area of Responsibility cannot be blank.");
        else
            this.area = area;
    }

    public String getSupervisorsName()
    {
        return this.supervisorsName;
    }

    public void setSupervisorsName(String supervisorsName)
    {
        if (supervisorsName == null)
            throw new NullPointerException("Supervisor's Name cannot be null.");
        else if (supervisorsName.equals(""))
            throw new IllegalArgumentException("Supervisor's Name cannot be blank.");
        else
            this.supervisorsName = supervisorsName;
    }

    public String getTitle()
    {
        return this.title;
    }

    public void setTitle(String title)
    {
        if (title == null)
            throw new NullPointerException("Title cannot be null.");
        else if (title.equals(""))
            throw new IllegalArgumentException("Title cannot be blank.");
        else
            this.title = title;
    }
    // <-- END GETTERS AND SETTERS

    /**
     * Equals method to determine equality between this Administrator object and another.
     * @param other The other Administrator object that will be checked for equality.
     * @return boolean, indicating whether this Administrator object is equal to <b>other</b>.
     */
    public boolean equals(Administrator other)
    {
        if (other == null)
```

```
            throw new NullPointerException();
        else
            return (super.equals(other) &&
                    this.getArea().equals(other.getArea()) &&
                    this.getSupervisorsName().equals(other.getSupervisorsName()) &&
                    this.getTitle().equals(other.getTitle()));
    }

    /**
     * Overridden toString method to serialize this object into string form.
     * @return String, representing this Administrator object in string form.
     */
    public String toString()
    {
        return (super.toString() + "\n" +
                this.getTitle() + " of " + this.getArea() + "\n" +
                "Supervised by " + this.getSupervisorsName());
    }

    /**
     * Interactive method to get information from keyboard input by the user.
     */
    public void readAdminInfo()
    {
        boolean valid = false;
        Scanner keyboardInput = new Scanner(System.in);

        // Keep trying until fully valid input is obtained.
        while (!valid)
        {
            try
            {
                System.out.println("Employee's Name:");
                this.setName(keyboardInput.nextLine());
                System.out.println("Employee's Date of Hire:");
                Date tempDate = new Date();
                tempDate.readInput();
                this.setHireDate(tempDate);
                System.out.println("Employee's Yearly Salary:");
                this.setSalary(Double.parseDouble(keyboardInput.nextLine()));
                System.out.println("Employee's Title:");
                this.setTitle(keyboardInput.nextLine());
                System.out.println("Employee's Area of Responsibility:");
                this.setArea(keyboardInput.nextLine());
                System.out.println("Employee's Supervisor:");
                this.setSupervisorsName(keyboardInput.nextLine());
                valid = true;
            }
            catch (Exception ex)
            {
                System.out.println("ERROR!");
                System.out.println(ex.getMessage() + "\n");
            }
        }
    }
}
```

Using the AdministratorDemo class to do test the default constructor, 6-parameter constructor, copy constructor, getters, equals method, readAdminInfo method, and toString method of the Administrator class. In this example, readAdminInfo has good input (default).

```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-Project\bin
>java AdministratorDemo
admin1: No name, January 1, 1000
$0.0 per year
No Title of No Area
Supervised by No Supervisor
admin2: Bob, February 3, 1999
$34000.0 per year
Vice President of Human Resources
Supervised by Joe
admin3 copy of admin2: Bob, February 3, 1999
$34000.0 per year
Vice President of Human Resources
Supervised by Joe
Get the title from admin2: Vice President
Get the area from admin2: Human Resources
Get the supervisors name from admin2: Joe
admin2 equal to admin3: true
admin1 equal to admin3: false
Employee's Name:
Dan
Employee's Date of Hire:
Enter month, day, and year.
Do not use a comma.
June 1 2015
Employee's Yearly Salary:
36000
Employee's Title:
Supervisor
Employee's Area of Responsibility:
Information Technology
Employee's Supervisor:
Bob Ace
admin1: Dan, June 1, 2015
$36000.0 per year
Supervisor of Information Technology
Supervised by Bob Ace

C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-Project\bin
>_
```

Showing that the readAdminInfo class won't take an empty string for the Title, Area of Responsibility, and Supervisor's Name.

```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-Project\bin
>java AdministratorDemo
admin1: No name, January 1, 1000
$0.0 per year
No Title of No Area
Supervised by No Supervisor
admin2: Bob, February 3, 1999
$34000.0 per year
Vice President of Human Resources
Supervised by Joe
admin3 copy of admin2: Bob, February 3, 1999
$34000.0 per year
Vice President of Human Resources
Supervised by Joe
Get the title from admin2: Vice President
Get the area from admin2: Human Resources
Get the supervisors name from admin2: Joe
admin2 equal to admin3: true
admin1 equal to admin3: false
Employee's Name:
Dan
Employee's Date of Hire:
Enter month, day, and year.
Do not use a comma.
June 1 2015
Employee's Yearly Salary:
36000
Employee's Title:

ERROR!
Title cannot be blank.

Employee's Name:
Dan
Employee's Date of Hire:
Enter month, day, and year.
Do not use a comma.
June 1 2015
Employee's Yearly Salary:
36000
Employee's Title:
Supervisor
Employee's Area of Responsibility:

ERROR!
Area of Responsibility cannot be blank.

Employee's Name:
Dan
Employee's Date of Hire:
Enter month, day, and year.
Do not use a comma.
June 1 2015
Employee's Yearly Salary:
36000
Employee's Title:
Supervisor
Employee's Area of Responsibility:
Information Technology
Employee's Supervisor:

ERROR!
Supervisor's Name cannot be blank.

Employee's Name:
Dan
Employee's Date of Hire:
Enter month, day, and year.
Do not use a comma.
June 1 2015
Employee's Yearly Salary:
36000
Employee's Title:
Supervisor
Employee's Area of Responsibility:
Information Technology
Employee's Supervisor:
Bob Ace
admin1: Dan, June 1, 2015
$36000.0 per year
Supervisor of Information Technology
Supervised by Bob Ace

C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW2\HW2-Project\bin
>_
```