```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)
```

```java
                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");
                }
            }
        }
    }
```