```java
import javax.swing.JOptionPane;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Random;

/**
 * A class that imitates a simple phone.
 *
 * This send/receives texts, streams audio, and charges
 * as well as keeping track of data usage.
 * It will print a monthly statement with all of the statistics and
 * bills for the phone.
 * <br>
 * Example code:<br>
 * {@code MyPhone phone = new MyPhone("John Doe", "0102030405");}<br>
 * {@code phone.chargeBattery(50);}<br>
 * {@code phone.streamAudio(100);}<br>
 * {@code phone.sendText("Hello");}<br>
 * {@code phone.readText();}<br>
 * {@code phone.printStatement();}<br>
 *
 * @author Silas Agnew
 * @version 1.0.0
 */
public class MyPhone
{
    private static final String DEFAULT_NUMBER = "9999999999";
    private static final double DATA_PER_MIN = 65 / 60.0;
    private static final double MAX_MINUTES = 720.0;

    private int textCount;
    private double dataConsumed;
    private double batteryLevel;
    private String name;
    private String number;

    //-Constructors-------------------------------------------------//

    /**
     * Constructs a phone
     * @param name Name of the phone user
     * @param number Phone number (must be all digits with a length of 10)
     */
    public MyPhone(String name, String number)
    {
        textCount = 0;
        dataConsumed = 0.0;
        batteryLevel = 0.0;
        setName(name);
        setPhoneNumber(number);
    }

    //-Accessors----------------------------------------------------//

    /**
     * @return Number of texts sent or received
     */
    public int getNumTexts() { return textCount; }

    /**
     * @return Percentage of battery life remaining
     */
    public double getBatteryLife() { return batteryLevel; }

    /**
     * @return Amount of data used in MB
     */
    public double getDataUsage() { return dataConsumed; }

    //-Mutators---------------------------------------------------//

    /**
     * @param name Name to set {@code MyPhone.name} to
     */
    public void setName(String name) { this.name = name; }

    /**
     * Sets the phone number, if valid, to {@code number}.
     * If the number is invalid it will set the phone number to
     * {@code DEFAULT_NUMBER} (9999999999)
     * @param number The phone number to set. Should be exactly 10 characters
     *               of only digits
     * @return if the passed in string was valid
     */
    public boolean setPhoneNumber(String number)
    {
        if (number.length() == 10)
        {
            this.number = number;
            return true;
        }
        else this.number = DEFAULT_NUMBER;
        return false;
    }

    /**
     * Charges the battery for the value that is equivalent to
     * charging it for time: {@code mins}
     * A full charge is 120 minutes so percentage of charge added is
     * found from {@code mins}/120
     * @param mins Metaphorical minutes to charge for
     */
    public void chargeBattery(int mins)
    {
        if (mins <= 0) return;
        batteryLevel = mins / 120.0 <= 1 - batteryLevel ?
                batteryLevel + mins / 120.0 : 1;

        JOptionPane.showMessageDialog(
                null, "Battery Level: " + (int)(batteryLevel * 100) + "%");
    }

    /**
     * Simulates streaming audio
     * This dissipates the battery and will kill the battery
     * if the streaming is for longer than the battery allows
     * @param mins Minutes to stream audio
     */
    public void streamAudio(int mins)
    {
        if (Double.compare(batteryLevel, 0) <= 0) return;
        double minutes = 0;
        if (Double.compare((double)mins, MAX_MINUTES * batteryLevel) > 0)
        {
            minutes = MAX_MINUTES * batteryLevel;
            JOptionPane.showMessageDialog(
                    null, "Warning: MyPhone runs out of battery after "
                            + minutes + " minutes");
        }
        else minutes = mins;

        batteryLevel -= minutes / MAX_MINUTES;
        dataConsumed += minutes * DATA_PER_MIN;
    }

    /**
     * Simulates sending a text and shows the sent text in a popup message
     * @param text Text to send
     */
    public void sendText(String text)
    {
        if (batteryLevel < 0.0001) return;
        textCount++;
        JOptionPane.showMessageDialog(null, "You: " + text);
    }

    /**
     * Receives a text
     * The text is randomly generated and displayed in a popup message
     */
    public void readText()
    {
        if (batteryLevel < 0.0001) return;

        Random rnd = new Random();
        int length = rnd.nextInt(4) + 25;
        char gen[] = new char[length];
        String text;

        for (int i = 0; i < length; i++)
        {
            gen[i] = (char)(rnd.nextInt(96) + 32);
        }
        text = new String(gen);

        switch (rnd.nextInt(5))
        {
            case 0:
                text += " Oh sry wrong person";
                break;
            case 1:
                text += " Who dis?";
                break;
            case 2:
                text += " K.";
                break;
            case 3:
                text += " Why are you texting me?";
                break;
            case 4:
                text += " Wat?";
                break;
        }

        textCount++;
        JOptionPane.showMessageDialog(null, "Rando: " + text);
    }

    /**
     * Prints out the monthly statement for the phone
     * This includes the customer info and usage stats as well as
     * the monthly bill
     * NOTE: Renews the month at the end of the call
     */
    public void printStatement()
    {
        NumberFormat fmt = NumberFormat.getCurrencyInstance();
        DecimalFormat dfmt = new DecimalFormat("#.##");

        System.out.println("MyPhone Monthly Statement\n");

        System.out.println("Customer: " + name);
        System.out.println("Number: " + fmtPhoneNumber());
        System.out.println("Texts: " + textCount);
        System.out.println("Data usage: " +
                dfmt.format(dataConsumed / 1000) + " (GB)\n");

        System.out.println("2GB Plan: " + fmt.format(50));
        System.out.println("Additional data fee: " +
                fmt.format(calcAdditionalDataFee()));
        System.out.println("Universal Usage (3%): " +
                fmt.format(calcUsageCharge()));
        System.out.println("Administrative Fee: " + fmt.format(0.61));
        System.out.println("Total Charges: " + fmt.format(calcTotalFee()));

        startNewMonth();
    }

    //-Helpers-----------------------------------------------------//

    /**
     * Resets usage data for a new measurement period
     */
    private void startNewMonth()
    {
        dataConsumed = 0.0;
        textCount = 0;
    }

    /**
     * @return Fee for over-use data in dollars
     */
    private double calcAdditionalDataFee()
    {
        double addDataUsedGB = dataConsumed / 1000 - 2;
        return addDataUsedGB < 0.0001 ? 0 : Math.ceil(addDataUsedGB) * 15;
    }

    /**
     * @return The usage charge in dollars
     */
    private double calcUsageCharge()
    {
        return (50 + calcAdditionalDataFee()) * 0.03;
    }

    /**
     * The total monthly fee is the sum of the service cost,
     * usage charge, and administrative fees
     * @return The total monthly fee in dollars
     */
    private double calcTotalFee()
    {
        return calcUsageCharge() + calcAdditionalDataFee() + 50 + .61;
    }

    /**
     * Formats the phone number into format: (xxx) xxx-xxxx
     * @return Formatted phone number
     */
    private String fmtPhoneNumber()
    {
        return "(" + number.substring(0, 3) + ") " +
                number.substring(3, 6) + "-" + number.substring(6, 10);
    }
}

```