```java
// 8.4

public class Rectangle {
    private double length = 1.0;
    private double width = 1.0;

    public Rectangle() {}

    public Rectangle(double length, double width) {
        setLength(length);
        setWidth(width);
    }

    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        if (length > 0.0 && length < 20.0) {
            this.length = length;
        } else {
            throw new IllegalArgumentException("Length must be between 0.0 and
20.0");
        }
    }

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        if (width > 0.0 && width < 20.0) {
            this.width = width;
        } else {
            throw new IllegalArgumentException("Width must be between 0.0 and
20.0");
        }
    }

    public double perimeter() {
        return 2 * (length + width);
    }

    public double area() {
        return length * width;

    }

    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 10);
        System.out.println("Length: " + rectangle.getLength());
        System.out.println("Width: " + rectangle.getWidth());
        System.out.println("Perimeter: " + rectangle.perimeter());
        System.out.println("Area: " + rectangle.area());
    }
}
```

```java
// 8.5

public class Time2 {
    private int totalSeconds;

    public Time2() {
        this(0, 0, 0);
    }

    public Time2(int h, int m, int s) {
        setTime(h, m, s);
    }

    public void setTime(int h, int m, int s) {
        totalSeconds = h * 3600 + m * 60 + s;
    }

    public int getHour() {
        return totalSeconds / 3600;
    }

    public int getMinute() {
        return (totalSeconds % 3600) / 60;
    }

    public int getSecond() {
        return totalSeconds % 60;
    }

    public String toUniversalString() {
        return String.format("%02d:%02d:%02d", getHour(), getMinute(),
getSecond());
    }

    public String toString() {
        return String.format("%d:%02d:%02d %s",
                (getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12,
                getMinute(), getSecond(), (getHour() < 12) ? "AM" : "PM");
    }

    public static void main(String[] args) {
        Time2 time = new Time2();
        System.out.println("The initial universal time is: " +
time.toUniversalString());
        System.out.println("The initial standard time is: " +
time.toString());
        System.out.println();

        time.setTime(13, 27, 6);
        System.out.println("Universal time after setTime is: " +
time.toUniversalString());
        System.out.println("Standard time after setTime is: " +
time.toString());
    }
}
```

```java
// 8.6

class Time1 {
    private int totalSeconds;

    public Time1() {
        this(0, 0, 0);
    }

    public Time1(int h, int m, int s) {
        setTime(h, m, s);
    }

    public void setTime(int h, int m, int s) {
        totalSeconds = h * 3600 + m * 60 + s;
    }

    public int getHour() {
        return totalSeconds / 3600;
    }

    public int getMinute() {
        return (totalSeconds % 3600) / 60;
    }

    public int getSecond() {
        return totalSeconds % 60;
    }

    public void tick() {
        totalSeconds++;
        if (totalSeconds ≥ 86400) { // 86400 seconds in a day
            totalSeconds = 0;
        }
    }
```

```java
    public void incrementMinute() {
        totalSeconds += 60;
        if (totalSeconds ≥ 86400) {
            totalSeconds = 0;
        }
    }

    public void incrementHour() {
        totalSeconds += 3600;
        if (totalSeconds ≥ 86400) {
            totalSeconds = 0;
        }
    }

    public String toUniversalString() {
        return String.format("%02d:%02d:%02d", getHour(), getMinute(), getSecond());
    }

    public String toString() {
        return String.format("%d:%02d:%02d %s",
                (getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12,
                getMinute(), getSecond(), (getHour() < 12) ? "AM" : "PM");
    }

    public static void main(String[] args) {
        Time1 time = new Time1(11, 59, 59);
        System.out.println("Initial time: " + time.toUniversalString());
        time.tick();
        System.out.println("Time after tick: " + time.toUniversalString());
    }
}
```

```java
// 8.7

public class Date {
    private int month;
    private int day;
    private int year;

    private static final int[] daysPerMonth =
    { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    public Date(int month, int day, int year) {
        if (month <= 0 || month > 12)
            throw new IllegalArgumentException("month (" + month + ") must be 1-12");

        if (day <= 0 ||
            (day > daysPerMonth[month] && !(month == 2 && day == 29)))
            throw new IllegalArgumentException("day (" + day +
                ") out-of-range for the specified month and year");

        if (year < 1800 || year > 9999)
            throw new IllegalArgumentException("year (" + year + ") must be 1800-9999");

        this.month = month;
        this.day = day;
        this.year = year;
    }
```

```
    public void nextDay() {
        if (day < daysPerMonth[month]) {
            day++;
        } else {
            if (month == 2 && day == 28 && !isLeapYear()) {
                day++;
            } else if (month == 12) {
                day = 1;
                month = 1;
                year++;
            } else {
                day = 1;
                month++;
            }
        }
    }

    public String toString() {
        return String.format("%d/%d/%d", month, day, year);
    }

    private boolean isLeapYear() {
        return year % 400 == 0 || (year % 4 == 0 && year % 100 ≠ 0);
    }

    public static void main(String[] args) {
        Date date = new Date(12, 31, 2020);
        System.out.println("Initial date: " + date);
        for (int i = 1; i ⩽ 365; i++) {
            date.nextDay();
            System.out.println("Next day: " + date);
        }
    }
}
```

```java
// 8.8

public class Time3 {
    private int hour;
    private int minute;
    private int second;

    public Time3() {
        this(0, 0, 0);
    }

    public Time3(int h, int m, int s) {
        setTime(h, m, s);
    }

    public boolean setHour(int h) {
        if (h ≥ 0 && h < 24) {
            hour = h;
            return true;
        } else {
            return false;
        }
    }

    public boolean setMinute(int m) {
        if (m ≥ 0 && m < 60) {
            minute = m;
            return true;
        } else {
            return false;
        }
    }

    public boolean setSecond(int s) {
        if (s ≥ 0 && s < 60) {
            second = s;
            return true;
        } else {
            return false;
        }
    }

    public boolean setTime(int h, int m, int s) {
        if (setHour(h) && setMinute(m) && setSecond(s)) {
            return true;
        } else {
            return false;
        }
    }
```

```java
    public int getHour() {
        return hour;
    }

    public int getMinute() {
        return minute;
    }

    public int getSecond() {
        return second;
    }

    public String toUniversalString() {
        return String.format("%02d:%02d:%02d", hour, minute, second);
    }

    public String toString() {
        return String.format("%d:%02d:%02d %s",
                ((hour == 0 || hour == 12) ? 12 : hour % 12),
                minute, second, (hour < 12) ? "AM" : "PM");
    }

    public static void main(String[] args) {
        Time3 time = new Time3();
        if (!time.setHour(25)) {
            System.out.println("Invalid hour value");
        }
        if (!time.setMinute(60)) {
            System.out.println("Invalid minute value");
        }
        if (!time.setSecond(61));
    }
}
```

```java
// 8.9

enum TrafficLight {
    RED(30), GREEN(45), YELLOW(5);

    private int duration;

    private TrafficLight(int duration) {
        this.duration = duration;
    }

    public int getDuration() {
        return duration;
    }

    public static void main(String[] args) {
        for (TrafficLight light : TrafficLight.values()) {
            System.out.printf("%s: %d seconds%n", light, light.getDuration());
        }
    }
}
```

```java
// 8.10

public class Complex {
    private double real;
    private double imaginary;

    public Complex() {
        this(0, 0);
    }

    public Complex(double real, double imaginary) {
        this.real = real;
        this.imaginary = imaginary;
    }

    public Complex add(Complex other) {
        return new Complex(real + other.real, imaginary + other.imaginary);
    }

    public Complex subtract(Complex other) {
        return new Complex(real - other.real, imaginary - other.imaginary);
    }

    public void print() {
        System.out.printf("(%.2f, %.2f)", real, imaginary);
    }
```

```
    public static void main(String[] args) {
        Complex c1 = new Complex(3, 2);
        Complex c2 = new Complex(1, 7);
        Complex c3 = c1.add(c2);
        Complex c4 = c1.subtract(c2);

        System.out.print("c1 = ");
        c1.print();
        System.out.print("\nc2 = ");
        c2.print();
        System.out.print("\nc1 + c2 = ");
        c3.print();
        System.out.print("\nc1 - c2 = ");
        c4.print();
    }
}
```

```java
// 8.11


class Time2 {
    private int hour;
    private int minute;
    private int second;

    public Time2() {
        this(0, 0, 0);
    }

    public Time2(int h, int m, int s) {
        setTime(h, m, s);
    }

    public boolean setHour(int h) {
        if (h ≥ 0 && h < 24) {
            hour = h;
            return true;
        } else {
            return false;
        }
    }
```

```java
public boolean setMinute(int m) {
    if (m ⩾ 0 && m < 60) {
        minute = m;
        return true;
    } else {
        return false;
    }
}

public boolean setSecond(int s) {
    if (s ⩾ 0 && s < 60) {
        second = s;
        return true;
    } else {
        return false;
    }
}

public boolean setTime(int h, int m, int s) {
    if (setHour(h) && setMinute(m) && setSecond(s)) {
        return true;
    } else {
        return false;
    }
}
```

```
public int getHour() {
    return hour;
}

public int getMinute() {
    return minute;
}

public int getSecond() {
    return second;
}

public void incrementHour() {
    if (++hour == 24) {
        hour = 0;
    }
}

public void incrementMinute() {
    if (++minute == 60) {
        minute = 0;
        incrementHour();
    }
}
```

```
    public void incrementSecond() {
        if (++second == 60) {
            second = 0;
            incrementMinute();
        }
    }
    public String toUniversalString() {
        return String.format("%02d:%02d:%02d", hour, minute, second);
    }

    public String toString() {
        return String.format("%d:%02d:%02d %s",
                ((hour == 0 || hour == 12) ? 12 : hour % 12),
                minute, second, (hour < 12 ? "AM" : "PM"));
    }
}

class Date {
    private int month;
    private int day;
    private int year;

    private static final int[] daysPerMonth =
    { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

```
    public Date() {
        this(1, 1, 1800);
    }

    public Date(int month, int day, int year) {
        if (month <= 0 || month > 12)
            throw new IllegalArgumentException("month (" + month + ") must be
1-12");

        if (day <= 0 ||
            (day > daysPerMonth[month] && !(month == 2 && day == 29)))
            throw new IllegalArgumentException("day (" + day +
                ") out-of-range for the specified month and year");

        if (year < 1800 || year > 9999)
            throw new IllegalArgumentException("year (" + year + ") must be
1800-9999");

        this.month = month;
        this.day = day;
        this.year = year;
    }

    public void nextDay() {
        if (day < daysPerMonth[month]) {
```

```
            day++;
        } else {
            if (month == 2 && day == 28 && isLeapYear()) {
                day++;
            } else if (month == 12 && day == 31) {
                day = 1;
                month = 1;
                year++;
            } else if (day == daysPerMonth[month]){
                day = 1;
                month++;
            }
        }
    }

    public boolean isLeapYear() {
        if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)) {
            return true;
        } else {
            return false;
        }
    }

    public String toString() {
        return String.format("%d/%d/%d", month, day, year);
```

```
    }
}

public class DateAndTime {
    private Date date;
    private Time2 time;

    public DateAndTime() {
        this(new Date(), new Time2());
    }

    public DateAndTime(Date date, Time2 time) {
        this.date = date;
        this.time = time;
    }

    public void incrementHour() {
        time.incrementHour();
        if (time.getHour() == 0) {
            date.nextDay();
        }
    }

    public void incrementMinute() {
        time.incrementMinute();
```

```
        if (time.getMinute() == 0 && time.getHour() == 0) {
            date.nextDay();
        }
    }

    public void incrementSecond() {
        time.incrementSecond();
        if (time.getSecond() == 0 && time.getMinute() == 0 && time.getHour()
== 0) {
            date.nextDay();
        }
    }

    public String toUniversalString() {
        return date.toString() + " " + time.toUniversalString();
    }

    public String toString() {
        return date.toString() + " " + time.toString();
    }

    public static void main(String[] args) {
        DateAndTime dateAndTime = new DateAndTime();
        System.out.println(dateAndTime.toUniversalString());
        System.out.println(dateAndTime.toString());
```

```
        dateAndTime.incrementSecond();
        System.out.println(dateAndTime.toUniversalString());
        System.out.println(dateAndTime.toString());

        dateAndTime.incrementMinute();
        System.out.println(dateAndTime.toUniversalString());
        System.out.println(dateAndTime.toString());

        dateAndTime.incrementHour();
        System.out.println(dateAndTime.toUniversalString());
        System.out.println(dateAndTime.toString());
    }
}
```

```java
// 8.12

class Enhanced_Rectangle {
    private double x1, y1, x2, y2, x3, y3, x4, y4;

    public Enhanced_Rectangle(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {
        setCoordinates(x1, y1, x2, y2, x3, y3, x4, y4);
    }

    public void setCoordinates(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {
        if (x1 < 0 || x2 < 0 || x3 < 0 || x4 < 0 || y1 < 0 || y2 < 0 || y3 < 0 || y4 < 0) {
            throw new IllegalArgumentException("All coordinates must be in the first quadrant");
        }

        if (x1 > 20 || x2 > 20 || x3 > 20 || x4 > 20 || y1 > 20 || y2 > 20 || y3 > 20 || y4 > 20) {
            throw new IllegalArgumentException("No single x- or y-coordinate can be larger than 20");
        }

        if ((x1 == x2 && y1 == y2) || (x1 == x3 && y1 == y3) || (x1 == x4 && y1 == y4) || (x2 == x3 && y2 == y3) || (x2 == x4
&& y2 == y4) || (x3 == x4 && y3 == y4)) {
            throw new IllegalArgumentException("The supplied coordinates do not form a Rectangle");
        }

        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
        this.x3 = x3;
        this.y3 = y3;
        this.x4 = x4;
        this.y4 = y4;
    }

    public double length() {
        double length1 = Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
        double length2 = Math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
        return Math.max(length1, length2);
    }

    public double width() {
        double width1 = Math.sqrt((x1 - x4) * (x1 - x4) + (y1 - y4) * (y1 - y4));
```

```
        double width2 = Math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3));
        return Math.min(width1, width2);
    }

    public double perimeter() {
        double length = length();
        double width = width();
        return 2 * (length + width);
    }

    public double area() {
        double length = length();
        double width = width();
        return length * width;
    }

    public boolean isSquare() {
        double length = length();
        double width = width();
        return length == width;
    }

    public static void main(String[] args) {
        Enhanced_Rectangle rect = new Enhanced_Rectangle(1,1,1,10,10,10,10,1);
        System.out.println("Length: " + rect.length());
        System.out.println("Width: " + rect.width());
        System.out.println("Perimeter: " + rect.perimeter());
        System.out.println("Area: " + rect.area());
        System.out.println("Is square: " + rect.isSquare());
    }

}
```

```java
// 8.13

public class IntegerSet {
    private boolean[] set;

    // no-argument constructor initializes the set to the "empty set"
    public IntegerSet() {
        set = new boolean[101];
    }

    // insert method adds an element to the set
    public void insert(int element) {
        if (element ≥ 0 && element ≤ 100) {
            set[element] = true;
        }
    }

    // delete method removes an element from the set
    public void delete(int element) {
        if (element ≥ 0 && element ≤ 100) {
            set[element] = false;
        }
    }

    // get method returns the element at a given index
    public boolean get(int element) {
        if (element ≥ 0 && element ≤ 100) {
            return set[element];
        }
        return false;
    }

    // union method returns a new set that is the union of the current set and another set
    public IntegerSet union(IntegerSet other) {
        IntegerSet result = new IntegerSet();
```

```
    for (int i = 0; i <= 100; i++) {
        if (this.get(i) || other.get(i)) {
            result.insert(i);
        }
    }
    return result;
}

// intersection method returns a new set that is the intersection of the current set and another set
public IntegerSet intersection(IntegerSet other) {
    IntegerSet result = new IntegerSet();
    for (int i = 0; i <= 100; i++) {
        if (this.get(i) && other.get(i)) {
            result.insert(i);
        }
    }
    return result;
}

// toString method returns a string representation of the set
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("{");
    for (int i = 0; i <= 100; i++) {
        if (set[i]) {
            sb.append(i + ",");
        }
    }
    if (sb.length() > 1) {
        sb.deleteCharAt(sb.length() - 1);
    }
    sb.append("}");
    return sb.toString();
}
```

```
    public static void main(String[] args) {
        IntegerSet set1 = new IntegerSet();
        set1.insert(1);
        set1.insert(2);
        set1.insert(3);
        System.out.println("Set 1: " + set1);

        IntegerSet set2 = new IntegerSet();
        set2.insert(2);
        set2.insert(3);
        set2.insert(4);
        System.out.println("Set 2: " + set2);

        IntegerSet union = set1.union(set2);
        System.out.println("Union: " + union);

        IntegerSet intersection = set1.intersection(set2);
        System.out.println("Intersection: " + intersection);
    }
}
```

```java
// 8.14 A


import java.text.SimpleDateFormat;
import java.util.Calendar;

public class NewDate {
    private Calendar calendar;

    public NewDate() {
        calendar = Calendar.getInstance();
    }

    public NewDate(int year, int month, int day) {
        calendar = Calendar.getInstance();
        calendar.set(year, month - 1, day);
    }

    public int getYear() {
        return calendar.get(Calendar.YEAR);
    }

    public int getMonth() {
        return calendar.get(Calendar.MONTH) + 1;
    }

    public int getDay() {
        return calendar.get(Calendar.DATE);
    }

    public String getFormattedDate() {
        SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");
        return sdf.format(calendar.getTime());
    }
```

```
    public String getFormattedDate2() {
        SimpleDateFormat sdf = new SimpleDateFormat("MMMM dd, yyyy");
        return sdf.format(calendar.getTime());
    }

    public String getFormattedDate3() {
        SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy");
        return sdf.format(calendar.getTime());
    }

    public static void main(String[] args) {
        NewDate date = new NewDate();
        System.out.println(date.getFormattedDate());
        System.out.println(date.getFormattedDate2());
        System.out.println(date.getFormattedDate3());
    }
}
```

```java
// 8.14 B

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class MyDate {
    private Calendar calendar;

    // no-arg constructor that initializes the date to the current date
    public MyDate() {
        calendar = Calendar.getInstance();
    }

    // constructor that initializes the date to a specified date in the format MM/DD/YYYY
    public MyDate(int month, int day, int year) {
        calendar = Calendar.getInstance();
        calendar.set(year, month - 1, day);
    }

    // constructor that initializes the date to a specified date in the format June 14, 1992
    public MyDate(String month, int day, int year) throws ParseException {
        calendar = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("MMMM dd, yyyy");
        java.util.Date date = sdf.parse(month + " " + day + ", " + year);
        calendar.setTime(date);
    }

    // constructor that initializes the date to a specified date in the format 14 June 1992
    public MyDate(int day, String month, int year) throws ParseException {
        calendar = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy");
        java.util.Date utilDate = sdf.parse(day + " " + month + " " + year);
        calendar.setTime(utilDate);
```

```
}

// getter for the year
public int getYear() {
    return calendar.get(Calendar.YEAR);
}

// getter for the month
public int getMonth() {
    return calendar.get(Calendar.MONTH) + 1;
}

// getter for the day
public int getDay() {
    return calendar.get(Calendar.DATE);
}

// method to output the date in the format MM/DD/YYYY
public String getFormattedDate() {
    SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");
    return sdf.format(calendar.getTime());
}

// method to output the date in the format June 14, 1992
public String getFormattedDate2() {
    SimpleDateFormat sdf = new SimpleDateFormat("MMMM dd, yyyy");
    return sdf.format(calendar.getTime());
}

// method to output the date in the format 14 June 1992
public String getFormattedDate3() {
    SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy");
    return sdf.format(calendar.getTime());
}
```

```
    public static void main(String[] args) throws ParseException {
        MyDate date1 = new MyDate(6, 14, 1992);
        System.out.println(date1.getFormattedDate());
        System.out.println(date1.getFormattedDate2());
        System.out.println(date1.getFormattedDate3());
        System.out.println();

        MyDate date2 = new MyDate("June", 14, 1992);
        System.out.println(date2.getFormattedDate());
        System.out.println(date2.getFormattedDate2());
        System.out.println(date2.getFormattedDate3());
        System.out.println();

        MyDate date3 = new MyDate(14, "June", 1992);
        System.out.println(date3.getFormattedDate());
        System.out.println(date3.getFormattedDate2());
        System.out.println(date3.getFormattedDate3());
    }
}
```

```java
// 8.15

public class HugeInteger {
    private int[] digits;

    public HugeInteger() {
        digits = new int[40];
    }

    public void input(String number) {
        if (number.length() > 40) {
            System.out.println("Error: Number is too large to fit in the array.");
        } else {
            for (int i = 0; i < number.length(); i++) {
                digits[i] = Integer.parseInt(String.valueOf(number.charAt(i)));
            }
        }
    }

    public void output() {
        for (int digit : digits) {
            System.out.print(digit);
        }
        System.out.println();
    }

    public void add(HugeInteger other) {
        for (int i = 0; i < digits.length; i++) {
            digits[i] += other.digits[i];
        }
    }

    public void subtract(HugeInteger other) {
```

```
    for (int i = 0; i < digits.length; i++) {
        digits[i] -= other.digits[i];
    }
}

public boolean isEqualTo(HugeInteger other) {
    for (int i = 0; i < digits.length; i++) {
        if (digits[i] ≠ other.digits[i]) {
            return false;
        }
    }
    return true;
}

public boolean isNotEqualTo(HugeInteger other) {
    return !isEqualTo(other);
}

public boolean isLessThan(HugeInteger other) {
    for (int i = 0; i < digits.length; i++) {
        if (digits[i] < other.digits[i]) {
            return true;
        } else if (digits[i] > other.digits[i]) {
            return false;
        }
    }
    return false;
}

public boolean isGreaterThan(HugeInteger other) {
    for (int i = 0; i > digits.length; i++) {
        if (digits[i] > other.digits[i]) {
            return true;
```

```
        } else if (digits[i] < other.digits[i]) {
            return false;
        }
    }
    return false;
}

public boolean isGreaterThanOrEqualTo(HugeInteger other) {
    return (isGreaterThan(other) || isEqualTo(other));
}

public boolean isLessThanOrEqualTo(HugeInteger other) {
    return (isLessThan(other) || isEqualTo(other));
}

public static void main(String[] args) {
    HugeInteger num1 = new HugeInteger();
    HugeInteger num2 = new HugeInteger();

    num1.input("12345678901234567890123456789O");
    num2.input("98765432109876543210987654321O");

    System.out.print("Number 1: ");
    num1.output();
    System.out.print("Number 2: ");
    num2.output();

    num1.add(num2);
    System.out.print("Number 1 + Number 2: ");
    num1.output();

    num1.subtract(num2);
    System.out.print("Number 1 - Number 2: ");
```

```
        num1.output();

        if (num1.isEqualTo(num2)) {
            System.out.println("Number 1 is equal to Number 2.");
        } else {
            System.out.println("Number 1 is not equal to Number 2.");
        }

        if (num1.isGreaterThan(num2)) {
            System.out.println("Number 1 is greater than Number 2.");
        } else {
            System.out.println("Number 1 is not greater than Number 2.");
        }

        if (num1.isLessThan(num2)) {
            System.out.println("Number 1 is less than Number 2.");
        } else {
            System.out.println("Number 1 is not less than Number 2.");
        }

        if (num1.isGreaterThanOrEqualTo(num2)) {
            System.out.println("Number 1 is greater than or equal to Number 2.");
        } else {
            System.out.println("Number 1 is not greater than or equal to Number 2.");
        }

        if (num1.isLessThanOrEqualTo(num2)) {
            System.out.println("Number 1 is less than or equal to Number 2.");
        } else {
            System.out.println("Number 1 is not less than or equal to Number 2.");
        }
    }
}
```