1. | Expression                          | Value |
---|------------------------------------|-------|
    | 1 * 2 * 3 + (4 - 5)                | 5     |
    | 28 % 4 + 18 % 5 % 5 + 9            | 12    |
    | 1000 * 2 + 18 / 2 / 2 * 2          | 2008  |
    | 1 / 10.0 + "1" + 17 * 2            | "0.1134" |
    | 0.25 * 2 - 0.5 + 1 / 2            | 0.0   |

2. The program produces the following output:

   I do not like green ham and eggs
   I do not like am sam and i
   I do not like sam i and am
   I do not like green eggs and ham

3. | Method Call                  | Output Produced |
---|-----------------------------|-----------------|
    | ifElseMystery(12, 45);      | 12 44           |
    | ifElseMystery(15, 5);       | 3 4             |
    | ifElseMystery(64, 8);       | 13 8            |
    | ifElseMystery(12, 12);      | 1 11            |
    | ifElseMystery(20, 7);       | 20 7            |
    | ifElseMystery(100, 5);      | 20 5            |

4. | Method Call | Output Produced |
---|-------------|-----------------|
    | mystery(5); | 1 2 3 5         |
    | mystery(3); | 1 2             |
    | mystery(7); | 1 2 3 5 8 13    |
    | mystery(0); | 1               |

5. | x > y | z > 0 | y % 2 == 0 |
---|------|------|-----------|
    | +-----|------|-----------|
    | Point A | sometimes | never | never     |
    | Point B | always     | sometimes | sometimes |
    | Point C | sometimes | always  | sometimes |
    | Point D | sometimes | always  | always    |
    | Point E | never      | sometimes | sometimes |
6. Two possible solutions are shown below.

```java
public static boolean testFairCoin(Scanner console) {
    int heads = 0;
    int total = 0;

    System.out.print("next flip? ");
    String flip = console.next();
    while (!flip.equals("done")) {
        if (flip.equals("heads")) {
            heads++;
        }
        total++;
        System.out.print("next flip? ");
        flip = console.next();
    }
    double pct = 100.0 * heads / total;
    System.out.println("was heads "+ pct + "% of the time");
    return (pct >= 45 && pct <= 55);
}

public static boolean testFairCoin(Scanner console) {
    int heads = 0;
    int tails = 0;

    String flip = ";
    while (!flip.equals("done")) {
        System.out.print("next flip? ");
        flip = console.next();
        if (flip.equals("heads")) {
            heads++;
        } else if (flip.equals("tails")) {
            tails++;
        }
    }
    double pct = (double) heads / (heads + tails);
    System.out.println("was heads "+ (pct * 100) + "% of the time");
    return (pct >= .45 && pct <= .55);
}
```
7. Two possible solutions are shown below.

```java
public static void busyDay(int numMeetings, Random rand) {
    int totalTime = 0;
    int longest = 0;

    for (int i = 0; i < numMeetings; i++) {
        int meeting = rand.nextInt(46) + 15;
        totalTime += meeting;
        longest = Math.max(longest, meeting);
        System.out.println("Scheduled new " + meeting + "-min meeting; " +
                          "total time now " + (totalTime / 60) + "h " +
                          (totalTime % 60) + "m");
    }
    System.out.println("Longest meeting was " + longest + " minutes");
}

public static void busyDay(int numMeetings, Random rand) {
    int minutes = 0;
    int hours = 0;
    int longest = 0;

    for (int i = 0; i < numMeetings; i++) {
        int meeting = rand.nextInt(46) + 15;
        minutes += meeting;
        if (minutes >= 60) {
            hours++;
            minutes -= 60;
        }
        longest = Math.max(longest, meeting);
        System.out.println("Scheduled new " + meeting + "-min meeting; " +
                          "total time now " + hours + "h " +
                          minutes + "m");
    }
    System.out.println("Longest meeting was " + longest + " minutes");
}
```
Three possible solutions are shown below.

```java
public static boolean isMonotonic(int n, boolean incr) {
    int next = n % 10;
    n /= 10;
    while (n > 0) {
        int curr = n % 10;
        if ((curr >= next && incr) || (curr <= next && !incr)) {
            return false;
        }
        next = curr;
        n /= 10;
    }
    return true;
}
```

```java
public static boolean isMonotonic(int n, boolean incr) {
    if (n < 10) {
        return true;
    }
    while (n > 9) {
        int curr = n % 10;
        int prev = n % 100 / 10;
        if ((curr >= prev && !incr) || (curr <= prev && incr)) {
            return false;
        }
        n /= 10;
    }
    return true;
}
```
public static boolean isMonotonic(int n, boolean incr) {
    if (incr) {
        int next = n % 10;
        n /= 10;
        while (n > 0) {
            int curr = n % 10;
            if (curr >= next) {
                return false;
            }
            next = curr;
            n /= 10;
        }
        return true;
    } else {
        int next = n % 10;
        n /= 10;
        while (n > 0) {
            int curr = n % 10;
            if (curr <= next) {
                return false;
            }
            next = curr;
            n /= 10;
        }
        return true;
    }
}