```java
/**
 *  Gets a command-line argument (int), and prints all the divisors of the given
number.
 */
public class Divisors {
    public static void main (String[] args) {
        int inputNum = Integer.parseInt(args[0]);

        for (int i=1; i <= inputNum; i++) {
            if ( inputNum % i == 0) {
                System.out.println(i);
            }
        }
    }
}
```

```
/**
 * Prints a given string, backward. Then prints the middle character in the
string.
 * The program expects to get one command-line argument: A string.
 */
public class Reverse {
    public static void main (String[] args){
        String inputTxt = args[0], outputTxt = "";

        for (int i = inputTxt.length() - 1; i >= 0; i--) {
            outputTxt = outputTxt + inputTxt.charAt(i);
        }

        int midIndex = inputTxt.length() / 2;
        char middle = outputTxt.charAt(midIndex);
        System.out.println(outputTxt);
        System.out.println("The middle character is " + middle);
    }
}
```

```java
/**
 *  Generates and prints random integers in the range [0,10),
 *  as long as they form a non-decreasing sequence.
 */
public class InOrder {
    public static void main (String[] args) {
        //generates a random integer
        int currentNum = (int) (10 * Math.random()),
            prevNum = 0;

        //prints currentNum, creates new random num
        //if the new number equals or higher, continue loop
        do {
            prevNum = currentNum;
            System.out.print(currentNum + " ");
            currentNum = (int) (10 * Math.random());
        } while (currentNum >= prevNum);
    }
}
```

```java
/**
 *  Gets a command-line argument (int), and chekcs if the given number is
perfect.
 */
public class Perfect {
    public static void main (String[] args) {
        //gets integer input
        int N = Integer.parseInt(args[0]),
                divisorSum = 1;
        //initializes response String
        String trueResponse = N + " is a perfect number since " + N + " = 1";

        //checks and sums all of N's divisors
        for (int i=2; i < N; i++) {
            if (N % i == 0) {
                trueResponse = trueResponse + " + " + i;
                divisorSum += i;
            }
        }

        //1 isn't prime so it isn't perfect
        if(N == divisorSum && N != 1)
            System.out.println(trueResponse);
        else
            System.out.println(N + " is not a perfect number");
    }
}
```

```java
/**
 *  Gets a command-line argument n (int), and prints an n-by-n damka board.
 */
public class DamkaBoard {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);

        //outer for: responsible for row amount
        for (int i=0; i < N; i++) {
            //inner for: prints each row individually
            for (int j=0; j < N; j++) {
                if ( i % 2 == 0)
                    System.out.print("* ");
                else
                    System.out.print(" *");
            }
            System.out.println();
        }
    }
}
```

```java
import java.util.Random;
/**
 *  Computes some statistics about families in which the parents decide
 *  to have children until they have at least one child of each gender.
 *  The program expects to get two command-line arguments: an int value
 *  that determines how many families to simulate, and an int value
 *  that serves as the seed of the random numbers generated by the program.
 *  Example usage: % java OneOfEachStats 1000 1
 */
public class OneOfEachStats {
    public static void main (String[] args) {
        // Gets the two command-line arguments
        int T = Integer.parseInt(args[0]);
        int seed = Integer.parseInt(args[1]);
        // Initailizes a random numbers generator with the given seed value
        Random generator = new Random(seed);

        int totalBodyCount = 0,
            famsOf2 = 0,
            famsOf3 = 0,
            famsOf4More = 0;

        for(int i=0; i < T; i++) {
            //flags for getting each gender
            boolean gotBoy = false,
                    gotGirl = false;

            int bodyCount = 0;

            //as long as we dont have both a boy
            //and a girl, continue making babies
            while (!gotBoy || !gotGirl) {
                if (generator.nextDouble() < 0.5) {
                    gotGirl = true;
                }
                else {
                    gotBoy = true;
                }

                bodyCount++;
                totalBodyCount++;
            }

            if (bodyCount == 2) famsOf2++;
            else if (bodyCount == 3) famsOf3++;
            else famsOf4More++;
            }
            double avg = (double) totalBodyCount / T;
            System.out.println("Average: " + avg + " children to get at least one
of each gender.");
            System.out.println("Number of families with 2 children: " + famsOf2);
            System.out.println("Number of families with 3 children: " + famsOf3);
            System.out.println("Number of families with 4 or more children: " +
famsOf4More);


            if (famsOf2 >= Math.max(famsOf3, famsOf4More)) {
                System.out.println("The most common number of children is 2.");
            } else {
                if (famsOf3 >= famsOf4More) {
                    System.out.println("The most common number of children is
3.");
                } else {
                    System.out.println("The most common number of children is 4
or more.");
                }
            }

    }
}
```