```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 n = Integer.parseInt(args[0]); // Gets a given integer
        for (int i = 1; i <= n; i++) {
            if (n % i == 0) {
                System.out.println(i); // Prints all the divisors of the given number.
            }
        }
    }
}
```

```java
/**
 * 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 str = args[0]; // Gets a string to reverse
        int n = str.length();
        for (int i = n - 1; i >= 0; i--) {
            System.out.print(str.charAt(i)); // Prints the string backward
        }
        System.out.println(); // Prints a new line
        System.out.println("The middle character is " + str.charAt((n - 1) / 2)); // Prints the middle character.
    }
}

```

```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) {
        int n = 0;
        int m = (int) (10 * Math.random()); // Generates the first random integer in the range [0,10)
        do {
            System.out.print(m + " "); // Prints the first random integer
            n = m;
            m = (int) (10 * Math.random());// Generates the next random integer in the range [0,10)
        } while (m >= n); // Checks if the next integer is not less than the last integer that was printed.
    }
}

```

```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]); // Gets the size of board to print
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (i % 2 == 0) {
                    System.out.print("* "); // Prints n times in row the
                    string "* "
                } else {
                    System.out.print(" *"); // Prints n times in row the
                    string " *"
                }
            }
            System.out.println(); // Skips to the next line
        }
    }
}
```

```java
/**
 *  Gets a command-line argument (int), and checks if the given number is perfect.
 */
public class Perfect {
    public static void main (String[] args) {
        int n = Integer.parseInt(args[0]); // Gets a given integer
        String str = n + " is a perfect number since " + n + " = 1";
        int sum = 0;
        for (int i = 1; i <= n / 2; i++) {
            if (n % i == 0){
                sum += i;
                if (i > 1){
                    str += " + " + i;
                }
            }
        }
        // Checks if the given number is perfect, and prints a suitable response
        if ((sum == n) && (n > 0)) {
            System.out.println(str);
        } else {
            System.out.println(n + " is not a perfect number");
        }
    }
}

```

```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]); // Gets the number of families to simulate
        int seed = Integer.parseInt(args[1]);
        // Initailizes a random numbers generator with the given seed value
        Random generator = new Random(seed);
        int totalChildren = 0; // Counts the total number of children that were generated in the simulation
        int b = 0; // b is a variable that represents the number of families with 2 children
        int c = 0; // c is a variable that represents the number of families with 3 children
        int d = 0; // d is a variable that represents the number of families with 4 children or more
        for (int i = 1; i <= T; i++) {
            int numOfBoys = 0;
            int numOfGirls = 0;
            // Generates new children until the family has at least one child of each gender.
            while (numOfBoys == 0 || numOfGirls == 0) {
                double rnd = generator.nextDouble();
                if (rnd < 0.5){
                    numOfBoys ++;
                } else {
                    numOfGirls++;
                }
            }
            int numOfChildren = numOfBoys + numOfGirls; // Counts the total number of children that were born in each family
            totalChildren += numOfChildren;
            if (numOfChildren == 2) {
                b++;
            } else if (numOfChildren == 3) {
                c++;
            } else {
                d++;
            }
        }
        double average = (totalChildren/(double) T); // Computes the average number of children to have at least one child of each gender.
        System.out.println("Average: " + average + " children to get at least one of each gender.");
        System.out.println("Number of families with 2 children: " + b);
        System.out.println("Number of families with 3 children: " + c);
        System.out.println("Number of families with 4 or more children: " + d);
        String mostCommon = "2"; // Saves the the most common number of
        children in a family as a string
        if ((c > b) || (d > b)) {
            if (d > c) {
                mostCommon = "4 or more";
            } else {
                mostCommon = "3";
            }
        }
        System.out.println("The most common number of children is " +
        mostCommon + ".");

        //// In the previous version of this program, you used a statement
        like:
        //// double rnd = Math.random();
        //// Where "rnd" is the variable that stores the generated random
        value.
        //// In this version of the program, replace this statement with:
        //// double rnd = generator.nextDouble();
        //// This statement will generate a random value in the range [0,1),
        //// just like you had in the previous version, except that the
        //// randomization will be based on the given seed.
        //// This is the only change that you have to do in the program.

    }
}
```