```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 num  = Integer.parseInt(args[0]);
        for (int i = 1; i <= num; i++) {
            if (num % i == 0)
                System.out.println (i);
        }
    }
}
```

```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 getS = args[0];
        for (int i = getS.length() - 1; i >= 0; i --) {
            System.out.print(getS.charAt(i));
        }
        System.out.println();
        System.out.println("The middle character is
"+getS.charAt ((getS.length() - 1)/2));
    }
}
```

```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 num1;
        int num2;
        int temp = (int) (Math.random() * 10);
        do{
            System.out.print(temp + " ");
            num1 = temp;
            num2 = (int) (Math.random() * 10);
            temp = num2;
        } while(num2 >= num1);
    }
}
```

```java
/**
 *  Gets a command-line argument (int), and chekcs if the
given number is perfect.
 */
public class Perfect {
    public static void main (String[] args) {
        int N  = Integer.parseInt(args[0]);
        String ifPerfect = N + " is a perfect number since " +
N + " = 1";
        int count = 1;
        for (int i = 2; i < N; i++) {
            if (N % i == 0){
                ifPerfect += " + " + i;
                count += i;
            }
        }
        if (N == count){
            System.out.println(ifPerfect);
        }
        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]);
        for (int i = 1; i <= n; i++){
            if(i%2 == 0){
                for (int j = 1; j <= n; j++){
                    System.out.print(" *");
                }
            } else {

                for (int j = 1; j <= n; j++){
                    System.out.print("* ");
                }
            }
            System.out.println();
        }
    }
}
```

```java
/**
 *  Simulates the formation of a family in which the parents
decide
 *  to have children until they have at least one child of
each gender.
 */
public class OneOfEach {
    public static void main (String[] args) {
        int countG = 0;
        int countB = 0;
        int sumC;
        String s = "";
        while (countG == 0 || countB == 0){
            if(Math.random() < 0.5){
                countG ++;
                s += "g ";

            }
            else{
                countB ++;
                s += "b ";
            }
        }
        sumC = countG + countB;
        s = s.trim();
        System.out.println(s);
        System.out.print("You made it... and you now have " +
sumC + " children.");
    }
}
```

```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 counter = 0, twoC = 0, threeC = 0,
        fourC = 0, CommoNum = 0;
        double rNum = 0, Avg = 0;
        char FirstGender = ' ', NextGender = ' ';
        for (int i = 1; i <= T; i++){
            rNum = generator.nextDouble();
            if (rNum > 0.5){
            FirstGender = 'b';
            } else {
                FirstGender = 'g';
            }
            NextGender = FirstGender;
            counter = 1;
            Avg ++;

            while (NextGender == FirstGender){
                rNum = generator.nextDouble();
                if (rNum > 0.5){
                    NextGender = 'b';
                } else {
                    NextGender = 'g';
                }
                counter ++;
                Avg ++;
            }
```

```
            if (counter == 2){
                twoC++;
            } else if (counter == 3){
                threeC++;
            } else {
                fourC++;
            }
        }

        Avg = Avg/T;
        CommoNum = Math.max(twoC, threeC);
        CommoNum = Math.max(CommoNum, fourC);

        System.out.println("Average: " + Avg + " children to
get at least one of each gender.");
        System.out.println("Number of families with 2
children: " + twoC);
        System.out.println("Number of families with 3
children: " + threeC);
        System.out.println("Number of families with 4 or more
children: " + fourC);

        if (CommoNum == fourC){
            System.out.println("The most common number of
children is 4 or more.");
        } else if (CommoNum == threeC){
        System.out.println("The most common number of children
is 3.");
        } else {
        System.out.println("The most common number of children
is 2.");
    }
}
}
```