```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 Word = args[0];
        String ReverseWord = "";
        char MiddleChar = Word.charAt((Word.length() - 1) /
2);

        for (int i = Word.length() - 1; i >= 0; i--){
            ReverseWord += Word.charAt(i);
        }
        System.out.println(ReverseWord);
        System.out.println("The middle character is " +
MiddleChar);

    }
}
```

```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 LastRND =(int)(Math.random()* 10);
        System.out.print(LastRND + " ");
        int NextRND =(int)(Math.random()* 10);

        while (NextRND >= LastRND){
            LastRND = NextRND;
            System.out.print(NextRND + " ");
            NextRND =(int)(Math.random()* 10);
        }
    }
}
```

```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 CandidateNum = Integer.parseInt(args[0]);
        String Sum = CandidateNum + " is a perfect number
since " + CandidateNum + " = 1";
        int counter = 1;

        for (int i = 2; i < CandidateNum; i++){
            if (CandidateNum % i == 0){
                counter += i;
                Sum += " + " + i;
            }
        }

        if (counter == CandidateNum){
            System.out.println(Sum);
        } else {
            System.out.println(CandidateNum + " 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) {

        double RandomNum = Math.random();
        char FirstGender = ' ', NextGender = ' ';
        int counter = 1;
        if (RandomNum > 0.5){
            FirstGender = 'b';
        } else {
            FirstGender = 'g';
        }
        System.out.print(FirstGender + " ");
        NextGender = FirstGender;

        while (NextGender == FirstGender){
            RandomNum = Math.random();
            if (RandomNum > 0.5){
                NextGender = 'b';
            }   else {
                NextGender = 'g';
            }
            System.out.print(NextGender + " ");
            counter ++;
        }
        System.out.println();
        System.out.println("you made it... and you have " +
counter + " childrens." );

    }
}
```

```java
/**
 *  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 one command-line argument: an
int value
 *  that determines how many families to simulate.
 */
public class OneOfEachStats1 {
    public static void main (String[] args) {

        int T = Integer.parseInt(args[0]);
        int counter = 0, TwoChil = 0, ThreeChil = 0,
FourMoreChil = 0, CommoNum = 0;
        double RandomNum = 0,  Avg = 0;
        char FirstGender = ' ', NextGender = ' ';

        for (int i = 1; i <= T; i++){

            RandomNum = Math.random();
            if (RandomNum > 0.5){
                FirstGender = 'b';
            } else {
                FirstGender = 'g';
            }
            NextGender = FirstGender;
            counter = 1;
            Avg ++;

            while (NextGender == FirstGender){
                RandomNum = Math.random();
                if (RandomNum > 0.5){
                    NextGender = 'b';
                }   else {
                    NextGender = 'g';
                }
                counter ++;
                Avg ++;
            }

            if (counter == 2){
                TwoChil++;
            } else if (counter == 3){
                ThreeChil++;
            } else {
                FourMoreChil++;
            }
        }
```

```
        Avg = Avg/T;
        CommoNum = Math.max(TwoChil, ThreeChil);
        CommoNum = Math.max(CommoNum, FourMoreChil);

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

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




    }
}
```

```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);

        //// 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.


        int counter = 0, TwoChil = 0, ThreeChil = 0,
FourMoreChil = 0, CommoNum = 0;
        double RandomNum = 0,  Avg = 0;
        char FirstGender = ' ', NextGender = ' ';

        for (int i = 1; i <= T; i++){

            RandomNum = generator.nextDouble();
            if (RandomNum > 0.5){
```

```
                FirstGender = 'b';
            } else {
                FirstGender = 'g';
            }
            NextGender = FirstGender;
            counter = 1;
            Avg ++;

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

            if (counter == 2){
                TwoChil++;
            } else if (counter == 3){
                ThreeChil++;
            } else {
                FourMoreChil++;
            }
        }

        Avg = Avg/T;
        CommoNum = Math.max(TwoChil, ThreeChil);
        CommoNum = Math.max(CommoNum, FourMoreChil);

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

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

```
            System.out.println("The most common number of
children is 2.");
        }
    }
}
```