```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) {
        //// Put your code here

        int num = Integer.parseInt(args[0]);

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

            if(num%i ==0){
                System.out.println(i);
            }

        }
    }
}
```

```java
/**
```

```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) {
        //// Put your code here
        String originalString = args[0];
        String reversedString = "";
        char midChar;

        for (int i = originalString.length() - 1; i >= 0; i--) {

            reversedString = reversedString + originalString.charAt(i);
        }

        if (originalString.length()%2 == 0){

            midChar = originalString.charAt(originalString.length()/2 - 1);
        }
        else {
            midChar = originalString.charAt(((originalString.length()-1)/2));
        }



        System.out.println(reversedString);
        System.out.println("The middle character is " + midChar);
    }
}
```

```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) {
        //// Write your code here

        int num1 = (int)(Math.random()*10);
        System.out.print(num1);

        int num2 = (int)(Math.random()*10);

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

    }
}
```

```java
/**
 * Gets a command-line argument (int), and chekcs if the given number is
```

```java
 * perfect.
 */
public class Perfect {
    public static void main(String[] args) {
        //// Put your code here

        int num = Integer.parseInt(args[0]);
        int sum = 1;
        String outputString = num + " is a perfect number since " + num + " = 1";

        //loop 1 less than the number
        for (int i = 2; i <num; i++) {

            if (num % i == 0) {
                sum += i;
                outputString = outputString + " + " + i;
            }

        }

        if (sum == num){
            System.out.println(outputString);
        }
        else{
            System.out.println(num + " 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) {
        //// Put your code here
        int n = Integer.parseInt(args[0]);
        String outpuString = "";

        for (int i = 1; i <= n; i++) {
            outpuString = "";
            for (int j = 1; j <= n; j++) {

                if (i % 2 != 0) { // i is odd
                    outpuString = outpuString + "* ";
                } else {
                    outpuString = outpuString + " *";
                }
            }

            System.out.println(outpuString);
        }

    }
}
```

```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) {
      //// Put your code here

      boolean girl = false;
      boolean boy = false;
      int count = 0;


      do {

         if(Math.random() < 0.5){
            System.out.print("g ");
            girl = true;
         }
         else{
            System.out.print("b ");
            boy = true;
         }
         count ++;

      }while (!boy || !girl);

      System.out.println("\nYou made it... and you now have "+count+" children.");
   }
}
```

```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) {
        //// Put your code here
        boolean girl = false;
        boolean boy = false;
        int childCount = 0;

        int T = Integer.parseInt(args[0]);
        double sum = 0;
        int familyOf2 = 0;
        int familyOf3 = 0;
        int familyOf4 = 0;

        int mode = 0;

        for (int i = 0; i < T; i++) {
            childCount = 0;
            boy = false;
            girl = false;

            do {
                double rnd = Math.random();
                if (rnd < 0.5) {
                    girl = true;
                } else {
                    boy = true;
                }
                childCount++;

            } while (!boy || !girl);

            if (childCount == 2) {
                familyOf2++;
            } else if (childCount == 3) {
                familyOf3++;
            } else {
                familyOf4++;
            }
            sum += childCount;
        }
```

```
        mode = Math.max(Math.max(familyOf2, familyOf3), familyOf4);

        System.out.println("Average: " + (sum / T) + " children to get at least one of each
gender.");
        System.out.println("Number of families with 2 children: " + familyOf2);
        System.out.println("Number of families with 3 children: " + familyOf3);
        System.out.println("Number of families with 4 or more children: " + familyOf4);
        System.out.println("The most common number of children is "
                + ((mode == familyOf2) ? "2." : (mode == familyOf3) ? "3." : "4."));

    }
}
```

```
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);

      boolean girl = false;
      boolean boy = false;
      int childCount = 0;

      double sum = 0;
      int familyOf2 = 0;
      int familyOf3 = 0;
      int familyOf4 = 0;

      int mode = 0;

      for (int i = 0; i < T; i++) {
         childCount = 0;
         boy = false;
         girl = false;

         do {
            double rnd = generator.nextDouble();
            if (rnd < 0.5) {
               girl = true;
            } else {
               boy = true;
            }
            childCount++;

         } while (!boy || !girl);

         if (childCount == 2) {
            familyOf2++;
         } else if (childCount == 3) {
            familyOf3++;
         } else {
```

```
                familyOf4++;
            }
            sum += childCount;
        }

        mode = Math.max(Math.max(familyOf2, familyOf3), familyOf4);

        System.out.println("Average: " + (sum / T) + " children to get at least one of each
gender.");
        System.out.println("Number of families with 2 children: " + familyOf2);
        System.out.println("Number of families with 3 children: " + familyOf3);
        System.out.println("Number of families with 4 or more children: " + familyOf4);
        System.out.println("The most common number of children is "
                + ((mode == familyOf2) ? "2." : (mode == familyOf3) ? "3." : "4."));

        //// 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.

    }
}
```