```
public class Divisors {
    public static void main(String[] args) {

        // This is the number for which we want to find the divisors.
        int number = Integer.parseInt(args[0]);

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

            // Checks if 'i' is a divisor of 'number'
            if (number % i == 0) {

                System.out.println(i);
            }
        }
    }
}
```

```
public class Reverse {
    public static void main(String[] args) {

        if (args.length > 0) {
            String input = args[0];
            String reversed = "";
            int i = input.length() - 1;

            while (i >= 0) {
                reversed += input.charAt(i);
                i--;
            }

            System.out.println(reversed);

            // Check if the length of the input string is odd
            if (input.length() % 2 != 0) {

                // If odd, print the middle character
                System.out.println("The middle character is " + input.charAt(input.length() / 2));
            } else {

                // If even, print the character before the exact middle
                System.out.println("The middle character is " + input.charAt((input.length() / 2) - 1));
            }
        }
    }
}
```

```
public class InOrder {
    public static void main(String[] args) {

        // Generate a random integer between 0 and 9
        int lastNumber = (int) (Math.random() * 10);
        System.out.print(lastNumber);

        int newNumber;
        do {

            // Generate a new random integer between 0 and 9
            newNumber = (int) (Math.random() * 10);

            // Check if the new number is greater or equal to the last number
            if (newNumber >= lastNumber) {

                // If so, print the new number followed by a space
                System.out.print(" " + newNumber);

                // Update the last number to be the new number
                lastNumber = newNumber;
            }
            // Continue the loop as long as the new number is greater than or equal to the last number
        } while (newNumber >= lastNumber);

        System.out.println();
    }
}
```

```
public class Perfect {
   public static void main(String[] args) {

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

      int sum = 0;
      StringBuilder divisors = new StringBuilder();
      for (int i = 1; i < N; i++) {
         // Check if 'i' is a divisor of N.
         if (N % i == 0) {
            sum += i;

            if (divisors.length() > 0) {
               divisors.append(" + ");
            }
            divisors.append(i);
         }
      }

      // Check if the sum of divisors is equal to N.
      if (sum == N) {

         // Print the result if N is a perfect number.
         System.out.println(N + " is a perfect number since " + N + " = " + divisors);
      } else {

         // Print the result if N is not a perfect number.
         System.out.println(N + " is not a perfect number");
      }
   }
}
```

```
public class DamkaBoard {
    public static void main(String[] args) {

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

        for (int row = 0; row < n; row++) {

            // Determine if the row should start with an asterisk or a space
            boolean startWithAsterisk = row % 2 == 0;

            for (int col = 0; col < n; col++) {

                // Print an asterisk or a space depending on the row and column positions
                if (startWithAsterisk) {
                    System.out.print("* ");
                } else {
                    System.out.print(" *");
                }
            }

            // Move to the next line after completing a row
            System.out.println();
        }
    }
}
```

```
public class OneOfEach {
    public static void main(String[] args) {
        boolean hasBoy = false;
        boolean hasGirl = false;
        int childrenCount = 0;

        // Loop until both a boy and a girl have been "born"
        while (!hasBoy || !hasGirl) {

            // Generate a random number and check if it's less than 0.5
            if (Math.random() < 0.5) {

                // If true, it's a boy
                System.out.print("b ");
                hasBoy = true;  // Set the flag indicating a boy has been born
            } else {

                // Otherwise, it's a girl
                System.out.print("g ");
                hasGirl = true;  // Set the flag indicating a girl has been born
            }
            childrenCount++; // Increment the children count
        }

        // Once both a boy and a girl have been born, print the total number of children
        System.out.println("\nYou made it... and you now have " + childrenCount + " children.");
    }
}
```

```
public class OneOfEachStats1 {
    public static void main(String[] args) {
        if (args.length < 1) {
            System.out.println("Usage: java OneOfEachStats1 <number of experiments>");
            return;
        }

        int T = Integer.parseInt(args[0]);
        int[] countArray = new int[T + 1];
        int totalChildren = 0;

        // Run experiments for the specified number of times.
        for (int i = 0; i < T; i++) {
             // Flags to check if a boy or a girl is born.
            boolean hasBoy = false;
            boolean hasGirl = false;
```

```
import java.util.Random;
public class OneOfEachStats {

   public static void main(String[] args) {
      if (args.length < 1) {
         System.out.println("Usage: java OneOfEachStats <number of experiments>");
         return;
      }

      int T = Integer.parseInt(args[0]);
      int seed = Integer.parseInt(args[1]);
      Random generator = new Random(seed);

      // Initialize variables to keep track of statistics
      int totalChildren = 0;
      int count2 = 0, count3 = 0, count4OrMore = 0;
      int mostCommonCount = 0, mostCommonValue = 0;

      // Loop over the number of experiments
      for (int i = 0; i < T; i++) {
         boolean hasBoy = false;
         boolean hasGirl = false;
         int childrenCount = 0;

         // Keep having children until there is at least one boy and one girl
         while (!hasBoy || !hasGirl) {
            if (generator.nextDouble() < 0.5) {
               hasBoy = true;
            } else {
               hasGirl = true;
            }
            childrenCount++;
         }

         // Update total number of children
         totalChildren += childrenCount;

         // Update counts for 2, 3, and 4 or more children
         if (childrenCount == 2) {
            count2++;
         } else if (childrenCount == 3) {
            count3++;
         } else if (childrenCount >= 4) {
            count4OrMore++;
         }

         // Update the most common number of children based on the current counts
         int currentCount = childrenCount == 2 ? count2 : (childrenCount == 3 ? count3 : count4OrMore);
         if (currentCount > mostCommonCount) {
            mostCommonCount = currentCount;
            mostCommonValue = childrenCount >= 4 ? 4 : childrenCount;
         }
      }

      // Calculate the average number of children per family
      double average = (double) totalChildren / T;

      // Output the results
```

```
        System.out.println("Average: " + average + " children to get at least one of each gender.");
        System.out.println("Number of families with 2 children: " + count2);
        System.out.println("Number of families with 3 children: " + count3);
        System.out.println("Number of families with 4 or more children: " + count4OrMore);
        System.out.println("The most common number of children is " +
                (mostCommonValue == 4 ? "4 or more" : mostCommonValue) + ".");
    }
}
```

```
        // Counter for the number of children in the current experiment.
        int childrenCount = 0;

        // Continue having children until both a boy and a girl are born.
        while (!hasBoy || !hasGirl) {

            // Simulate the birth of a child with 50% chance for each gender.
            if (Math.random() < 0.5) {
                hasBoy = true;
            } else {
                hasGirl = true;
            }
            childrenCount++;
        }

        // Add the number of children from this experiment to the total.
        totalChildren += childrenCount;
        if (childrenCount <= T) {
            countArray[childrenCount]++;
        } else {
            countArray[T]++;
        }
    }

    // Calculate and print the average number of children to get at least one of each gender.
    double average = (double) totalChildren / T;
    System.out.println("Average: " + average + " children to get at least one of each gender.");

    // Print the number of families with exactly 2 or 3 children.
    System.out.println("Number of families with 2 children: " + countArray[2]);
    System.out.println("Number of families with 3 children: " + countArray[3]);

    // Calculate and print the number of families with 4 or more children.
    int fourOrMore = 0;
    for (int i = 4; i <= T; i++) {
        fourOrMore += countArray[i];
    }
    System.out.println("Number of families with 4 or more children: " + fourOrMore);

    // Find the most common number of children.
    int maxCount = 0;
    int mostCommon = 0;
    for (int i = 2; i <= T; i++) {
        if (countArray[i] > maxCount) {
            maxCount = countArray[i];
            mostCommon = i;
        }
    }
    System.out.println("The most common number of children is " +
            (mostCommon == T ? "4 or more" : mostCommon) + ".");
  }
}
```