```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 number = Integer.parseInt(args[0]);

        for (int i = 1; i <= number; i++) {
            if ((number % 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 originalString = args[0];
        String reversedString = "";
        int stringLength = originalString.length();

        for (int i = stringLength - 1; i >= 0; i--) {
            reversedString += originalString.charAt(i);
        }
        System.out.println(reversedString);

        int middleCharIndex = (stringLength - 1) / 2;
        char middleChar = originalString.charAt(middleCharIndex);
        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 randomNumber = (int) (Math.random() * 10);
        int previousNumber = -1;

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

```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 sideLength = Integer.parseInt(args[0]);

        for (int i = 1; i <= sideLength; i++) {
            boolean isEvenLine = (i % 2 == 0);

            for (int j = 0; j < sideLength; j++) {
                if (isEvenLine) {
                    System.out.print(" *");
                } else {
                    System.out.print("* ");
                }
            }
            System.out.println();
        }
    }
}
```

```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 number = Integer.parseInt(args[0]);
        int sumOfDivisors = 1; // 1 is divisor of every number
        String perfectNumberMessage = number + " is a perfect number since " + number + " = 1";

        for (int i = 2; i < number; i++) {
            if ((number % i) == 0) {
                sumOfDivisors += i;
                perfectNumberMessage += " + " + i;
            }
        }

        if (number == sumOfDivisors && number != 1) {
            System.out.println(perfectNumberMessage);
        } else {
            System.out.println(number + " 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) {
        int numberOfExperiments = 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 allChildrenCount = 0;
        int couplesWith2Children = 0;
        int couplesWith3Children = 0;
        int couplesWith4OrMoreChildren = 0;

        for (int i = 0; i < numberOfExperiments; i++) {
            boolean isBoyBorn = false;
            boolean isGirlBorn = false;
            int childrenPerCouple = 0;

            while (!isBoyBorn || !isGirlBorn) {
                double randomGender = generator.nextDouble();

                if (randomGender > 0.5) {
                    isBoyBorn = true;
                } else {
                    isGirlBorn = true;
                }

                childrenPerCouple++;
```

```
            }

            if (childrenPerCouple == 2) {
                couplesWith2Children++;
            } else if (childrenPerCouple == 3) {
                couplesWith3Children++;
            } else {
                couplesWith4OrMoreChildren++;
            }

            allChildrenCount += childrenPerCouple;
        }

        double averageChildrenPerCouple = (double) allChildrenCount / numberOfExperiments;
        System.out.println("Average: " + averageChildrenPerCouple + " children to get at least one of each gender.");

        System.out.println("Number of families with 2 children: " + couplesWith2Children);
        System.out.println("Number of families with 3 children: " + couplesWith3Children);
        System.out.println("Number of families with 4 or more children: " + couplesWith4OrMoreChildren);

        String mostCommonNumberOfChildrenMessage = "The most common number of children is ";
        int mostCommonNumberOfChildren = Math.max(couplesWith2Children,
                Math.max(couplesWith3Children, couplesWith4OrMoreChildren));

        if (mostCommonNumberOfChildren == couplesWith2Children) {
            mostCommonNumberOfChildrenMessage += "2.";
        } else if (mostCommonNumberOfChildren == couplesWith3Children) {
            mostCommonNumberOfChildrenMessage += "3.";
        } else {
            mostCommonNumberOfChildrenMessage += "4 or more.";
        }

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