```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 inputNumber = Integer.parseInt(args[0]);
        int d = 1;
        while (d <= inputNumber) {
            if ((inputNumber % d) == 0) {
                System.out.println(d);
            }
            d++;
        }
    }
}
```

```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 s = args[0];

        int stringLength = s.length();
        // Loop through the entire string length
        for (int i = (stringLength - 1); i >= 0; i--) {
            System.out.print(s.charAt(i));
        }
        // checks if the string lenhth is even or odd to find the middle character
        if ((stringLength % 2) == 0) {
            int midIndex = (stringLength - 1) / 2;
            System.out.println();
            System.out.println("The middle character is " + s.charAt(midIndex));
        } else {
            int midIndex = stringLength / 2;
            System.out.println();
            System.out.println("The middle character is " + s.charAt(midIndex));
        }
    }
}
```

```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) {
        // Setting the upper bound and creating and printing the first random number
        int upperBound = 10;
        int currentNumber = (int) (Math.random() * upperBound);

        System.out.print(currentNumber + " ");

        // as long as the last number that was printed is less than the upperbound
        // the loop keeps running
        while (currentNumber < upperBound) {
            // generating the next number
            int nextNumber = (int) (Math.random() * upperBound);

            // if the the next number and the last number form a non-decreasing
            // sequence it prints the next number
            // if not, it breaks the loop by setting the last number to the upper bound
            if (nextNumber >= currentNumber) {
                System.out.print(nextNumber + " ");
                currentNumber = nextNumber;
            } else {
                currentNumber = upperBound;
            }
        }
    }
}
```

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

        // Sets an initial string
        String outStr = inputNum + " is a perfect number since " +
                        inputNum + " = 1";
        int d = 1;
        int sumCheck = 1;
        while (d < inputNum) {
            if (((inputNum % d) == 0) && (d != 1)) {
                outStr = outStr + " + " + d ;
                sumCheck = sumCheck + d;
            }
            d++;
        }
        if (sumCheck == inputNum) {
            System.out.println(outStr);
        } else {
            System.out.println(inputNum + " 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 inputNum = Integer.parseInt(args[0]);

        int line = 1;
        System.out.println();

        while (line <= inputNum) {
            int len = 1;
            // if the line is even the line will start with " "
            while (len <= inputNum) {
                if ((line % 2) == 0) {
                    System.out.print(" *");
                } else {
                    System.out.print("* ");
                }
                len++;
            }
            System.out.println();
            line++;
        }
    }
}
```

```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);

        // Setting all of the outer-scope variables
        double totalTnums = 0;
        int familyOfTwo = 0;
        int familyOfThree = 0;
        int familyOfFour = 0;

        // Running the loop T times
        for (int i = 1; i <= T; i++) {

            // Setting the inner-scope vairables
            int boysNum = 0;
            int girlsNum = 0;
            int totalNum = 0;
            while ((boysNum < 1) || (girlsNum < 1)) {
                double boyOrGirl = generator.nextDouble();
                if (boyOrGirl < 0.5) {
                    boysNum++;
                } else {
                    girlsNum++;
                }
            }
            // calculate in each experiment the number of children
            totalNum = boysNum + girlsNum;

            // divides the number of children into the required groups
            if (totalNum == 2) familyOfTwo++;
            else if (totalNum == 3) familyOfThree++;
            else if (totalNum >= 4) familyOfFour++;

            totalTnums = totalTnums + totalNum;
        }
```

```
    System.out.println("Average: " + (totalTnums / T) + " children to get at least one of
                       each gender");
    System.out.println("Number of families with 2 children: " + familyOfTwo);
    System.out.println("Number of families with 3 children: " + familyOfThree);
    System.out.println("Number of families with 4 or more children: " + familyOfFour);

    // calculate the most common number of children
    if ((familyOfTwo > familyOfThree) && (familyOfTwo > familyOfFour)) {
        System.out.println("The most common number of children is 2.");
    } else if ((familyOfThree > familyOfTwo) && (familyOfThree > familyOfFour)) {
        System.out.println("The most common number of children is 3.");
    } else if ((familyOfFour > familyOfTwo) && (familyOfFour > familyOfThree)) {
        System.out.println("The most common number of children is 4 or more.");
    } else if ((familyOfThree > familyOfTwo) && (familyOfThree == familyOfFour)) {
        System.out.println("The most common number of children is 3.");
    } else if ((familyOfTwo > familyOfFour) && (familyOfThree == familyOfTwo)) {
        System.out.println("The most common number of children is 2.");
    } else if ((familyOfTwo == familyOfFour) && (familyOfThree == familyOfTwo)) {
        System.out.println("The most common number of children is 2.");
    System.out.println();
    }
    }
}
```