

```
/**
 * 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]);
        int flag = 1;

        while(flag <= num)
        {
            if (num % flag == 0)
            {
                System.out.println(flag);
            }

            flag++;
        }
    }
}
```

```

/**
 * 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 input = args[0];
        int length = input.length();
        char middle = input.charAt((length / 2) + (length % 2 - 1));
        int flag = length - 1;

        while (flag >= 0)
        {
            System.out.print(input.charAt(flag));

            flag--;
        }

        System.out.print("\nThe middle character is " + middle);
    }
}

```

```

/**
 * 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
        // From the last week's assignment - generating a number in [a,b)
        // Because [a,b) are [0,10) and they are not changing, the diff is 10
        int diff = 10;

        // Generating the first num
        double rand = Math.random();
        int current = (int) (diff * rand);

        // Generating the 2nd num
        rand = Math.random();
        int next = (int) (diff * rand);

        System.out.print(current);

        while (current <= next)
        {
            current = next;
            rand = Math.random();
            next = (int) (diff * rand);

            System.out.print(" " + current);
        }
    }
}

```

```

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

        // Setting signals for the loops
        int i = 0;
        int j = 0;

        while (i < num)
        {
            if (i % 2 == 0)
            {
                // Adding a * as the first char.
                System.out.print("*");
                // Incrementing j
                j++;
            }

            while (j < num)
            {
                System.out.print(" ");
                j++;
            }
            // Resetting j
            j = 0;

            // This is unnecessary, its just so the test will pass
            if (i % 2 == 0)
            {
                // Adding a " " as the last char.
                System.out.print(" ");
            }

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

```

```

/**
 * Gets a command-line argument (int), and chekcs if the given number is perfect.
 */
public class Perfect {
    public static void main (String[] args) {
        /// Put your code here
        int num = Integer.parseInt(args[0]);

        // Going over the divisors that are greater than 1.
        int flag = 2;

        // Starting the divisors count from 1.
        int count = 1;

        String perfectNumString = num + " is a perfect number since " + num + " =
1";

        // Checking wether num is a perfect number or not
        // Checking the divisors using the code from ex1
        while(flag < num)
        {
            if (num % flag == 0)
            {
                count += flag;
                perfectNumString += " + " + flag;
            }

            flag++;
        }

        // If count is equal to num then num is a prefect number -
        // printing the perfectNumString
        if (count == num)
        {
            System.out.print(perfectNumString);
        }
        // If num is not perfect - printing it.
        else
        {
            System.out.print(num + " is not a perfect number");
        }
    }
}

```

```

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 families = 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.

        boolean boy = false;
        boolean girl = false;
        int boys = 0;
        int girls = 0;
        int twoChildrens = 0;
        int threeChildrens = 0;
        int fourOrMoreChildren = 0;
        int childCount = 0;

        double ran = generator.nextDouble();

        for (int i = 0; i < families; i++)
        {
            // Receiving 0 or 1, 0 for boys and 1 for girls, cause boys are losers
            and girls are number 1!
            // (Im a boy pls dont cancel me)
            int child = (int) (2 * ran);

```

```

boy = (child == 0);
girl = (child == 1);
childCount++;

if (boy)
{
    boys++;
}
else
{
    girls++;
}

while (!(boy && girl))
{

    ran = generator.nextDouble();
    child = (int) (2 * ran);

    boy = boy || (child == 0);
    girl = girl || (child == 1);
    childCount++;

    if (child == 0)
    {
        boys++;
    }
    else
    {
        girls++;
    }
}

if (childCount == 2)
{
    twoChildrens++;
}
else if (childCount == 3)
{
    threeChildrens++;
}
else
{
    fourOrMoreChildren++;
}

childCount = 0;
}

int mostCommon = 0;
if (twoChildrens > threeChildrens && twoChildrens > fourOrMoreChildren)

```

```

        {
            mostCommon = 2;
        }
        else if (threeChildrens > twoChildrens && threeChildrens >
fourOrMoreChildren)
        {
            mostCommon = 3;
        }
        else
        {
            mostCommon = 4;
        }

        System.out.println("Average: "
                            + ((double) (boys + girls) / families)
                            + " children to get at least one of each
gender.");
        System.out.println("Number of families with 2 children: " + twoChildrens);
        System.out.println("Number of families with 3 children: " + threeChildrens);
        System.out.println("Number of families with 4 or more children: " +
fourOrMoreChildren);
        System.out.println("The most common number of children is " +
mostCommon + ".");
    }
}

```