```java
/**================================================================================
 **                            Ordered.java
 *? Gets three command-line arguments (int values). If the values are strictly
 *? ascending or strictly descending, prints true. Otherwise prints false.
 *@argument x int
 *@argument y int
 *@argument z int
 *@print True / False
 *================================================================================**/
public class Ordered {
    public static void main (String[] args) {
        // Declaring the variables
        int x = Integer.parseInt(args[0]);
        int y = Integer.parseInt(args[1]);
        int z = Integer.parseInt(args[2]);
        // Check if the variables are in strictly ascending or descending order
        if ((x < y && y < z) ||  (x > y && y > z)){
            System.out.println("True");
        }
        else {
            System.out.println("False");
        }
    }
}
/*============================ END OF Ordered.java ============================*/
```

```java
/**=========================================================================
**                           Reverse.java
*? The program expects to get one command-line argument: A string.
*? Prints a given string, backward. Then prints the middle character in the string.
*@argument str String
*@print Reversed string of str
*@print Middle char in str
*=========================================================================**/
public class Reverse {
    public static void main (String[] args){
        // Declaring the variable
        String str = args[0];
        String reversedString = "";
        // Reverse the string and find the middle character
        for (int i=str.length()-1; i >= 0; i--) {
            reversedString += str.charAt(i);
        }
        System.out.println(reversedString);
        if (str.length() % 2 == 0){
            char middleChar = str.charAt((int) (str.length() / 2) - 1);
            System.out.println("The middle character is " + middleChar);
        }
        else {
            char middleChar = str.charAt((int)(str.length() / 2));
            System.out.println("The middle character is " + middleChar);
        }
    }
}
/*============================ END OF Reverse.java ============================*/
```

```java
/**================================================================================
 **                            DamkaBoard.java
 *?  Gets a command-line argument n, and prints an n-by-n damka board.
 *@argument n int
 *@print A damka board
 *================================================================================**/
public class DamkaBoard {
    public static void main(String[] args) {
        // Declare the variable
        int n = Integer.parseInt(args[0]);
        // Printing the n-by-n board
        if (n < 0) {
            System.out.println("Invalid input please with a positive number");
        }
        else {
            for (int i = 0; i < n; i++){
                for (int j = 0; j < n; j++) {
                    if (i % 2 != 0) {
                    System.out.print(" *");
                    }
                    else {
                        System.out.print("* ");
                    }
                }
                System.out.println();
            }
        }
    }
}
/*============================ END OF DamkaBoard.java ============================*/
```

```java
/**================================================================================
 **                            Perfect.java
 *? Gets a command-line argument, and chekcs if the given number is perfect.
 *@argument number int
 *@print If perfect: The number with its divisors. If not perfect: number is not perfect
 *================================================================================**/
public class Perfect {
    public static void main (String[] args) {
        // Declaring the variable
        int number = Integer.parseInt(args[0]);
        int sum = 0;
        String msg = number + " is a perfect number since " + number + " =";
        // Calculating and sum the divisors of the given number
        if (number>0){
            for(int divisor=1; divisor<number; divisor++) {
                if (number % divisor == 0) {
                    sum += divisor;
                    if (divisor == 1) {
                        msg += " " + divisor;
                    }
                    else{
                        msg += " + " + divisor;
                    }
                }
            }
        }
        else {
            for(int divisor=-1; divisor>number; divisor--) {
                if (number % divisor == 0) {
                    sum += divisor;
                    if (divisor == -1) {
                        msg += " (" + divisor + ")";
                    }
                    else{
                        msg += " + (" + divisor + ")";
                    }
                }
            }
        }
        if (sum == number){
        System.out.println(msg);
        }
        else {
            System.out.println(number + " is not a perfect number");
        }
    }
}
/*============================ END OF Perfect.java ============================*/
```

```java
import java.util.Random;
/**=========================================================================
 **                          OneOfEachStats.java
 *?  Computes some statistics about families in which the parents decide
 *?  to have children until they have at least one child of each gender.
 *@param T int
 *@param seed Int
 *@print Average of children in a family
 *@print Number of families that has two children
 *@print Number of families that has three children
 *@print Number of families that has four or more children
 *@print Most common number of children in a family
 *=========================================================================**/
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]);
        // Initializes a random numbers generator with the given seed value
        Random generator = new Random(seed);

        // Declaring the variables
        int hasTwoChildren = 0;
        int hasThreeChildren = 0;
        int hasFourPlusChildren = 0;
        double totalChildrenNum = 0;

        // Simulating the families and make the calculation
        if (T <= 0){
            System.out.println("Families number is invalid. Please try again with a positive integer.");
        }
        else {
            for (int i=0; i<T; i++) {
                int boy=0;
                int girl=0;
                while (boy < 1 || girl < 1) {
                    double child = generator.nextDouble();
                    if(child <= 0.5){
                        boy ++;
                    }
                    else {
                        girl ++;
                    }
                }
                totalChildrenNum += (boy + girl);
                if ((boy + girl) == 2){
                    hasTwoChildren++;
                }
                else if ((boy + girl) == 3) {
                    hasThreeChildren++;
                }
                else {
                    hasFourPlusChildren++;
                }
            }

            // Print the results
            System.out.println("Average: " + (totalChildrenNum/T) + " children to get at least one of each
gender.");
            System.out.println("Number of families with 2 children: " + hasTwoChildren);
            System.out.println("Number of families with 3 children: " + hasThreeChildren);
            System.out.println("Number of families with 4 or more children: " + hasFourPlusChildren);

            if (hasTwoChildren >= hasThreeChildren && hasTwoChildren >= hasFourPlusChildren) {
                    System.out.println("The most common number of children is 2.");
            }
            else if (hasThreeChildren >= hasFourPlusChildren && hasThreeChildren >= hasTwoChildren) {
                    System.out.println("The most common number of children is 3.");
            }
            else {
                    System.out.println("The most common number of children is 4.");
            }
        }
    }
}
/*=========================== END OF OneOfEachStats.java ===========================*/
```