```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 boardSize = Integer.parseInt(args[0]);
        String astStrart = "";
        String spaceStart = "";

        for (int i=0; i<boardSize * 2; i++) {
            if (i % 2 == 0) {
                astStrart += "* ";
            } else {
                spaceStart += " *";
            }
        }

        for (int i=0; i<boardSize; i++) {
            System.out.println(i % 2 == 0 ? astStrart : spaceStart);
        }
    }
}
```

```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);

        double numberOfExperiments = Double.parseDouble(args[0]);
        int tryNum = 1;
        int sumOfAllKids = 0;
        int parentsWith2kids = 0;
        int parentsWith3kids = 0;
        int parentsWith4orMorekids = 0;

        while (tryNum <= numberOfExperiments) {
            double numOfGirls = 0;
            double numOfBoys = 0;

            while (numOfGirls == 0 || numOfBoys == 0 || numOfBoys < 1 || numOfGirls < 1) {
                double theRandom = generator.nextDouble();
                String boyOrGirl = theRandom > 0.5 ? "b" : "g";

                if (boyOrGirl == "b") {
                    numOfBoys++;
                } else {
                    numOfGirls++;
                }
            }

            double numOfKids = numOfGirls + numOfBoys;

            if (numOfKids == 2) {
                parentsWith2kids++;
            } else if (numOfKids == 3) {
                parentsWith3kids++;
            } else if (numOfKids >= 4) {
                parentsWith4orMorekids++;
            }

            sumOfAllKids += numOfKids;
            tryNum++;
        }

        double average = sumOfAllKids / numberOfExperiments;
        String message2 = "Average: " + average + " children to get at least one of each gender.";
        System.out.println(message2);

        String message3 = "Number of families with 2 children: " + parentsWith2kids;
        System.out.println(message3);

        String message4 = "Number of families with 3 children: " + parentsWith3kids;
        System.out.println(message4);

        String message5 = "Number of families with 4 or more children: " + parentsWith4orMorekids;
        System.out.println(message5);

        String mostCommon = "";

        if (parentsWith2kids >= parentsWith3kids) {
            if (parentsWith2kids >= parentsWith4orMorekids) {
                mostCommon = "2";
            } else {
                mostCommon = "4 or more";
            }
        } else {
            if (parentsWith3kids >= parentsWith4orMorekids) {
                mostCommon = "3";
            } else {
                mostCommon = "4 or more";
            }
        }

        String message6 = "The most common number of children is " + mostCommon + ".";
        System.out.println(message6);
    }
}
```

```java

/**
 *  Gets three command-line arguments (int values). If the values are strictly
 *  ascending or strictly descending, prints true. Otherwise prints false.
 */
public class Ordered {
    public static void main (String[] args) {
        int first = Integer.parseInt(args[0]);
        int second = Integer.parseInt(args[1]);
        int third = Integer.parseInt(args[2]);

        Boolean isAscending = third > second && second > first;
        Boolean isDescending = third < second && second < first;

        if (isAscending == true || isDescending == true) {
            System.out.println("true");
        } else {
            System.out.println("false");
        }
    }
}
```

```java
/**
 *  Gets a command-line argument (int), and chekcs if the given number is perfect.
    (30 points) A number is said to be perfect if it equals the sum of all its divisors. For example, the
    divisors of 6 are 1, 2, and 3, and 6 = 1 + 2 + 3. Therefore 6 is a perfect number. Write a program
    ( perfect.java ) that takes an integer command-line argument value, say N, and checks if the
    number is perfect. Here are some examples of the program's execution:

    Test your program on, at least, the following numbers: 6, 24, 28, 496, 5002, 8128. Hint: four of
    these numbers are perfect. You can find a list of perfect numbers in the Internet, and use your
    program to verify that some of them are indeed perfect.

    Implementation tips: We suggest the following strategy. When you get a number, say 24, start
    by building the string " 24 is a perfect number since 24 = 1 ". Next, enter a loop that looks for
    all the divisors of 24. This loop can be identical to what you did in the Divisors program. When
    you find a divisor, append " + " and this divisor to the end of the string. At the end of the loop,
    you will know if 24 is indeed a perfect number. If so, print the string that you've constructed all
    along. If 24 is not a perfect number, ignore the string that you've constructed and print instead
    " 24 is not a perfect number ".
 */
public class Perfect {
    public static void main (String[] args) {
        int numToInspect = Integer.parseInt(args[0]);
        int divisor = 1;
        int divisorsSum = 0;
        String divisorsSumCalculation = "";

        while (numToInspect > divisor) {
            if (numToInspect % divisor == 0) {
                divisorsSum = divisorsSum + divisor;

                String stringedDivisor = "" + divisor;

                if (divisorsSumCalculation == "") {
                    divisorsSumCalculation = divisorsSumCalculation + stringedDivisor;
                } else {
                    divisorsSumCalculation = divisorsSumCalculation + " + " + stringedDivisor;
                }
            }

            divisor++;
        }

        String message = "";

        if (numToInspect == divisorsSum) {
            message = numToInspect + " is a perfect number since " + numToInspect + " = " + divisorsSumCalculation;
        } else {
            message = numToInspect + " is not a perfect number";
        }

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

```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.
    Use the string functions str.length() and str.charAt( i ) . You can read
    about them by consulting the String class API (search the Internet for “ java 16 string ”). The
    program can be implemented using either a for loop that goes backward, or a while loop that
    goes backward. Implement the program using a for loop. Then write a second implementation
 */
public class Reverse {
    public static void main (String[] args){
        String stringToCheck = args[0].toString();
        String reveredString = "";
        String middleChar = "";

        for (int i = stringToCheck.length() - 1; i >= 0; i--) {
            reveredString = reveredString + stringToCheck.charAt(i);
        }

        if (stringToCheck.length() % 2 == 0) {
            middleChar = middleChar + stringToCheck.charAt((stringToCheck.length() / 2) - 1);
        } else {
            middleChar = middleChar + stringToCheck.charAt((stringToCheck.length() / 2));
        }

        System.out.println(reveredString);
        String message = "The middle character is " + middleChar;
        System.out.println(message);
    }
}
```