Divisors.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) {
                continue;
            }

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

Reverse.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 stringToReverse = args[0];
        String reversedString = "";

        // Build the reversed string
        for (int i = stringToReverse.length() - 1; i >= 0; --i) {
            reversedString += stringToReverse.charAt(i);
        }

        System.out.println(reversedString);
        System.out.println("The middle character is "
                + reversedString.charAt(reversedString.length() / 2));
    }
}
```

InOrder.java

```
import java.util.Random;

/**
 * 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) {

        Random randomObject = new Random();

        int generatedNumber = randomObject.nextInt(10);
        int previousNumber;
        do {
            System.out.print(generatedNumber + " ");
            previousNumber = generatedNumber;
            generatedNumber = randomObject.nextInt(10);
        } while (generatedNumber >= previousNumber);
    }
}
```

DamkaBoard.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]);

        for (int i = 0; i < boardSize; ++i) {
            if (i % 2 == 0) {
                // repeat is pretty sick I love it
                System.out.println("* ".repeat(boardSize));
            } else {
                System.out.println(" *".repeat(boardSize));
            }
        }
    }
}
```

Perfect.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 numberToTest = Integer.parseInt(args[0]);
        String positiveResult =
                String.format("%d is a perfect number since %d = 1",
                        numberToTest, numberToTest);

        // add 1 since we already have it
        int sum = 1;

        for (int i = 2; i < numberToTest; ++i) {
            if (numberToTest % i > 0) {
                continue;
            }

            positiveResult += String.format(" + %d", i);
            sum += i;
        }

        if (sum == numberToTest) {
            System.out.println(positiveResult);
        } else {
            System.out.printf("%d is not a perfect number",
                    numberToTest);
        }
    }
}
```

OneOfEachStats.java

```
import java.util.Random;

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);

        int familiesWithTwoChildren = 0;
        int familiesWithThreeChildren = 0;
        int familiesWithFourOrMoreChildren = 0;
        String mostCommon;
        Double totalChildCount = 0.0;

        for (int i = 0; i < T; ++i) {
            Boolean boyCreated = false;
            Boolean girlCreated = false;
            int childrenCount = 0;

            while (!boyCreated || !girlCreated) {
                if (generator.nextDouble() <= 0.5) {
                    boyCreated = true;
                } else {
                    girlCreated = true;
                }
                childrenCount++;
            }

            totalChildCount += childrenCount;

            if (childrenCount >= 4) {
                familiesWithFourOrMoreChildren++;
            } else if (childrenCount == 3) {
                familiesWithThreeChildren++;
            } else {
                familiesWithTwoChildren++;
            }
        }

        System.out.println("Average: " + totalChildCount / T
                + " children to get at least one of each gender.");
```

```
        System.out.printf(
                "Number of families with 2 children: %d\n",
                familiesWithTwoChildren);
        System.out.printf(
                "Number of families with 3 children: %d\n",
                familiesWithThreeChildren);
        System.out.printf(
                "Number of families with 4 or more children: %d\n",
                familiesWithFourOrMoreChildren);

        if (familiesWithFourOrMoreChildren >
familiesWithThreeChildren) {
            mostCommon =
                    familiesWithFourOrMoreChildren >
familiesWithTwoChildren
                            ? "4 or more"
                            : "2";
        } else {
            mostCommon =
                    familiesWithThreeChildren >
familiesWithTwoChildren
                            ? "3"
                            : "2";
        }
        System.out.printf(
                "The most common number of children is %s.\n",
                mostCommon);
    }
}
```