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 num = Integer.parseInt(args[0]);
        for(int i =1;i<=num;i++){
            if (num % i == 0){
                System.out.println(i);
            }
        }
    }
}
```

Reverse

```
/**
 * 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 word = args[0];
        String ans = "";
        char midChar = 'd';
        for(int i = 0; i < word.length();i++)
        {
                ans += word.charAt(word.length()-i-1);
            if (i == (word.length()-1)/2){
                midChar= word.charAt(i);
            }
        }
        /*  int i = 0;
        while (i < word.length()){
            ans += word.charAt(word.length()-i-1);
            if (i == (word.length()-1)/2){
                midChar= word.charAt(i);
            }
         }
        */
        System.out.println(ans+"\nThe middle character is
"+midChar);
    }
}
```

InOrder

```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) {
        int rand;
        int num1 = 0;
        do{
            rand = (int)(Math.random()*10);

            if (num1 > rand){
                num1 = 10;
            }
            else{
                System.out.println(rand);
                num1 = rand;
            }
        }while(rand >= num1);
    }
}
```

DamkaBoard

```
/**
 *  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 size = Integer.parseInt(args[0]);
        for(int i = 0; i < size; i++){
            for(int j = 0; j < size; j++){
                if (i % 2 == 0)
                System.out.print("* ");
                else
                System.out.print(" *");
            }
            System.out.println();
        }
    }
}
```

Perfect

```
/**
 *  Gets a command-line argument (int), and chekcs if the given
number is perfect.
 */
public class Perfect {
    public static void main (String[] args) {
        int num = Integer.parseInt(args[0]);
        int sum = 1;
        String answer = num+" is a perfect number since "+num+" =
1";
        for (int i = 2; i <= num/2; i++){
            if (num % i == 0){
                answer+=" + "+i;
                sum += i;
            }
        }
        if (num == sum){
            System.out.println(answer);
        }
        else{
            System.out.println(num+" is not a perfect number");
        }
    }
}
```

OneOfEachStats

```
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);
        int familyOfTwo = 0;
        int familyOfThree = 0;
        int familyOfFourPlus = 0;
        Double average = 0.0;
        for (int i = 0; i <T; i++ )
        {
            int childCount = 0;
            boolean boy = false;
            boolean girl = false;
            while( boy == false || girl == false )
            {
                double rand = generator.nextDouble();
                if (rand >= 0.5)
                {
                    childCount++;
                    boy = true;
                    //System.out.print("b ");
                }
                else
                {
                    childCount++;
                    girl = true;
                    //System.out.print("g ");
                }
            }
```

```
            average += childCount;
            switch (childCount) { // Adds to the static counter
                case 2:
                    familyOfTwo++;
                    break;
                case 3:
                    familyOfThree++;
                    break;
                default:
                    familyOfFourPlus++;
                    break;
            }
        }
        average /= T;
        System.out.println("Average: " + average + " 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: "+familyOfFourPlus);
        // showing the correct message according to the category of
families
        if (familyOfTwo > familyOfFourPlus){
            if (familyOfThree > familyOfTwo){
                System.out.println("The most common number of
children is 3.");
            }
            else{
                System.out.println("The most common number of
children is 2.");
            }
        }
        else{
            if (familyOfFourPlus > familyOfThree){
                System.out.println("The most common number of
children is 4 or more.");
            }
            else{
                System.out.println("The most common number of
children is 3.");
            }
        }
    }
}
```