Divisors

```
public class Divisors {
	public static void main (String[] args) {
		int n = Integer.parseInt(args[0]);
		for (int i = 1; i <= n; i++) {
			if ((n % i) == 0) {
				System.out.println(i);
			}
		}
	}
}
```

Reversed

```
public class Reverse {
	public static void main (String[] args){
		String word = args[0];
		String reversed = "";
		int length = word.length();
		char middle = word.charAt((length-1)/2);
		for (int i = 0; i < length; i++) {
			reversed = reversed + word.charAt(length-1-i);
		}
		System.out.println(reversed);
		System.out.println("The middle character is " + middle);
	}
}
```

InOrder

```
public class InOrder {
	public static void main (String[] args) {
		int random = 0;
		int new_random = 0;
		boolean flag = true;
		do {
			new_random = (int) (Math.random()*10);
			if (new_random >= random) {
				System.out.print(new_random + " ");
				random = new_random;
			} else {
				flag = false;
			}
		} while (flag);
		System.out.println();
	}
}
```

Perfect

```
public class Perfect {
	public static void main (String[] args) {
		int n = Integer.parseInt(args[0]);
		int sum = 1;
		String result = n + " is a perfect number since " + n + " = 1";
		// runs on all possible divisors of n
		for (int i = 2; i <= (n/2); i++) {
			// determines divisors and concatenates + sums accordingly
			if (n % i == 0) {
				result = result + " + " + i;
				sum = sum + i;
			}
		}
		// checks if the sum of the divisors equals the number
		if (n == sum) {
			System.out.println(result);
		} else {
			System.out.println(n + " is not a perfect number");
		}

	}
}
```

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 n = Integer.parseInt(args[0]);
		// loops run on rows (i) and coloumns (j) and prints the symbol * n times
		for (int i = 0; i < n; i++) {
			for (int j=0; j < (n-1); j++) {
				System.out.print("* ");
			}
			if (i % 2 == 0) { // adds beginning space for every other row until the
			last damka row
				System.out.print("* ");
				System.out.println();
				System.out.print(" ");
			} else {
				System.out.print("*");
				System.out.println();
			}
		}
	}
}
```

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 countTwo = 0;
                int countThree = 0;
                int countFourPlus = 0;
                int total = 0; // assigning variable for total number of children
                double average = 0; // assigning variable for average number of children
                per experiment
                for (int i = 0; i < T; i++) { // checks every experiment
                        int children = 0;
                        boolean boy = false;
                        boolean girl = false;
                        while (boy == false || girl == false) { // runs until both genders have
                        been born
                                if (generator.nextDouble() < 0.5) {
                                        boy = true;
                                } else {
                                        girl = true;
                                }
                                children++; // counts total children for experiment
                        }
                // incriminates total children and children categories according to number
                of children in each exepriment
                if (children == 2) countTwo++;
                else if (children ==3) countThree++;
```

```
                else countFourPlus++;
                total = total + children;
                }
                average = (double) total / T;
                System.out.println("Average: " + average + " children to get at least one of
                each gender.");
                System.out.println("Number of families with 2 children: "+ countTwo);
                System.out.println("Number of families with 3 children: "+ countThree);
                System.out.println("Number of families with 4 or more children: "+
                countFourPlus);
                // checks the most common number of children and prints
                int common = Math.max(countTwo, Math.max(countThree,
                countFourPlus));
                if (common == countTwo) {
                        System.out.println("The most common number of children is 2.");
                } else if (common == countThree) {
                        System.out.println("The most common number of children is 3.");
                } else {
                        System.out.println("The most common number of children is 4 or
                        more.");
                }
        }
}
```