```
/**
 *  Gets a command-line argument (int), and prints all the divisors of the given number.
 */
public class Divisors {
	public static void main (String[] args) {

		//insert valid integer input into variable num
		int num = Integer.parseInt(args[0]);

		//checking if i is a divisor of num. if so, print i
		for(int i = 1; i <= num; i++) {
			if(num%i==0){
				System.out.println(i);
			}
		}
	}
}
```

```
/**
 * 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){

		//// insert string input into variable s
		String s = args[0];

		// iterate through s characters backwards and prints them
		for(int I = s.length() - 1 ; I >= 0 ; i--) {
			System.out.print(s.charAt(i));
		}
		// new line
		System.out.println();
		// print middle chararcter of string s.
		// if there even number of chars in s, print the last char of the first half
		System.out.println("The middle character is " + s.charAt((s.length() - 1) / 2));
	}
}
```

```
/**
 *  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) {

		//variable newNum stores a new random number in range each time
		int newNum  = (int)(Math.random()*10);
		//lastNum stores the last random number that was printed
		int lastNum = -1;

		//do while loop which runs at least one time
		do {
			/* print space between numbers in such a way that there is no
			space after the last number and not before the first number */
			if(lastNum != -1) System.out.print(" ");

			System.out.print(newNum);
			//implement newNum into lastNum
			lastNum = newNum;
			//creating new random number
			newNum = (int)(Math.random() * 10);

		      }while(newNum >= lastNum); //checking if newNum is now greater or equal to
                                                   the last number that was printed
		System.out.println();
	}
}
```

```
/**
 *  Gets a command-line argument n (int), and prints an n-by-n damka board.
 */
public class DamkaBoard {
	public static void main(String[] args) {
		//insert integer input from user into variable n
		int n = Integer.parseInt(args[0]);
		//iterate for each row
		for(int i = 0 ; i < n ; i++) {
			//iterate for each * in a row
			for(int j = 0 ; j < n ; j++) {
				//checking if the row should start with * or space
				if(I % 2 == 0)
					System.out.print("* ");
				else
					System.out.print(" *");
			}
			//making a new line
			System.out.println();
		}
	}
}
```

```
/**
 *  Gets a command-line argument (int), and chekcs if the given number is perfect.
 */
public class Perfect {
	public static void main (String[] args) {
		//insert integer input from user into variable num
		int num = Integer.parseInt(args[0]);
		//creating possitive outcome message
		String outcome_message = num + " is a perfect number since " + num + " = 1";
		int divisors_sum = 1;

		//iterate to check numbers from 2 to num-1 to see if they divide num
		for(int i = 2 ; i < num ; i++) {
			if(num%i==0) {
				//adding the new divisor to the final message and to divisors_sum
				outcome_message += " + " + i;
				divisors_sum += i;
			}
		}
		if(num == divisors_sum)
			System.out.println(outcome_message);
		else
			System.out.println(num + " is not a perfect number");
	}
}
```

```
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.
 */
public class OneOfEachStats {
        public static void main (String[] args) {

                int families_count = 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 total_children_count = 0;
                int families_count_with_2 = 0;
                int families_count_with_3 = 0;
                int families_count_with_4plus = 0;

                //creating x number of families. x = families_count
                for(int i=0 ; i<families_count ; i++) {

                        //setting local variables for each new family
                        boolean is_boy = false;
                        boolean is_girl = false;
                        int children_count = 0;
                        while(!is_boy || !is_girl) {
                                if(generator.nextDouble()>0.5) {
                                        is_boy = true;
                                }
                                else {
                                        is_girl = true;
                                }
                                children_count++;
                        }
                        total_children_count += children_count;
                        if(children_count==2)
                                families_count_with_2++;
                        else if(children_count==3)
                                families_count_with_3++;
                        else
                                families_count_with_4plus++;
                }
                //compute the average number of children per family
                double avg_children_in_family = total_children_count / (double)families_count;
                System.out.println("Average: " + avg_children_in_family + " children to get
                                        at least one of each gender.");
```

```
            System.out.println("Number of families with 2 children: " + families_count_with_2);
            System.out.println("Number of families with 3 children: " + families_count_with_3);
            System.out.println("Number of families with 4 or more children: "
                                    + families_count_with_4plus);


            //checking the most common number of children per family
            if(families_count_with_2>=families_count_with_3) {
                  if(families_count_with_2>=families_count_with_4plus)
                        System.out.println("The most common number of children is 2.");
                  else
                        System.out.println("The most common number of children is 4 or more.");
            }
            else if(families_count_with_3>=families_count_with_4plus)
                  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.");

      }
}
```