```
/**
 *  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 input = Integer.parseInt(args[0]);

		for (int i = 1; i <= input; i++){
			if ((input % 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){

          //Get input
          String input = args[0];

          String output = "";
          int i = input.length() - 1;

          //Reverse index for every letter in input
          while (i >= 0) {
               output += input.charAt(i);
               i--;
          }

          //Get the middle letter
          char middleChar = output.charAt(output.length()/2);

          //Print
          System.out.println(output);
          System.out.println("The middle character is " + middleChar);
```

```
        }
    }
```

```
/**
 *  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 random = 0;
          int previous = 0;
          boolean go = true;

          do {
               previous = random;
               random = (int)(Math.random() * 10);
               go = (previous <= random);
               if (go) System.out.print(random + " ");
          }

          while (go);
          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) {

		int input = Integer.parseInt(args[0]);

		for (int i = 0; i < input; i++){

			for (int j = 0; j < input; j++){
				if ((i % 2) == 0) System.out.print("* ");
				else System.out.print  (" *");
			}
			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) {

		String input  = args[0];
		int checkIfPerfect = Integer.parseInt(input);
		String isPerfect = input + " is a perfect number since " + input + " = 1";
		int countSumDivisors = 1;

		int i = 2;
		while (i < checkIfPerfect) {
			if ((checkIfPerfect % i) == 0) {
				isPerfect += " + " + i;
				countSumDivisors += i;
			}
			i++;
		}

		if (countSumDivisors == checkIfPerfect) System.out.println(isPerfect);
		else System.out.println(input + " 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.

 *  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 countFamiliesTwo = 0;

          int countFamiliesThree = 0;

          int countFamiliesMore = 0;

          double countChildrenAll = 0;
```

```
for (int i = 0; i < T; i++){

      boolean g = false;
      boolean b = false;
      int countChildren = 0;

      //Playing Sims in java
      do {
            double random = generator.nextDouble();
            if (random < 0.5)g = true;
            else b = true;
            countChildren += 1;
      }
      while ((g && b) == false);

      //Add to stats
      if (countChildren == 2) countFamiliesTwo += 1;
      else if (countChildren == 3) countFamiliesThree += 1;
      else countFamiliesMore += 1;
      countChildrenAll += countChildren;

}

//Calculates the average
double countChildrenAverage = (countChildrenAll / T);

//Detects which option is mode
```

```
            String whoIsMode;

            int mode = Math.max(Math.max(countFamiliesTwo,
countFamiliesThree), countFamiliesMore);

            if (mode == countFamiliesTwo) whoIsMode = "The most common
number of children is 2.";

            else if (mode == countFamiliesThree) whoIsMode = "The most
common number of children is 3.";

            else whoIsMode = "The most common number of children is 4 or
more.";



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

            System.out.println("Number of families with 2 children: " +
countFamiliesTwo);

            System.out.println("Number of families with 3 children: " +
countFamiliesThree);

            System.out.println("Number of families with 4 or more
children: " + countFamiliesMore);

            System.out.println(whoIsMode);


      }

}
```