Untitled-1

```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]);
        int index=1;
        while(index<=num)
        {
            if(num%index==0) System.out.println(index);
            index++;
        }
    }
}
```

```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 str = args[0];
        int length = str.length();
        // Print every char in the word from end to beginning
        for (int i=length-1; i>=0; i--)
        {
            System.out.print(str.charAt(i));
        }
        // check whether the length of the word is even or odd and print the middle char
        if(length%2==0)
        {
            System.out.println("\n"+ "The middle character is " + str.charAt((length/2)-1)
);
        }
        else System.out.println("\n"+ "The middle character is " + str.charAt(length/2));
    }
}
```

```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 first = (int)((Math.random()*10)); //generates the first random number
        int temp=first;
        System.out.print(first); // print the first random number
        int ran=temp;
        /* if (and only if) the new random number greater than the previous number
        generates another random number and print*/
        do {
            // Generates random numbers between [0-10)
            ran = (int)((Math.random()*10));
            if(ran>=temp)
            {
                System.out.print(" "+ran);
                temp=ran;
```

```
            }
        } while (ran>=temp);
    }
}
-----------------------------------------------------------------------------------------
/**
 *  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 number = Integer.parseInt(args[0]);
        for(int i=1; i<=number; i++) // i represent the number of the rows
        {
            String str = "";
            for(int j=1; j<=number; j++) // j represent the number of * in any row
            {
                str = str + "* ";
            }
            // prints the board - odd rows without space before the *
            if(i%2!=0) {System.out.println(str);}
            else System.out.println(" " + str.substring(0,str.length()-1));
        }
    }
}
-----------------------------------------------------------------------------------------
/**
 *  Gets a command-line argument (int), and chekcs if the given number is perfect.
 */
public class Perfect {
    public static void main (String[] args) {
        int number = Integer.parseInt(args[0]);
        int index=2;
        int sum=1; // start from 1 because 1 is a divisor of any number
        String str=number + " is a perfect number since " + number + " = 1"  ;
        // Find all divisors of the number and add to the string
        while(index<number)
        {
            if(number%index==0)
            {
                sum=sum+index;
                str = str + " + " + index;
            }
            index++;
        }
        //Check whether the sum of all divisors equals the number and print the result
        if (sum==number)
        {
            System.out.println(str);
        }
        else
        System.out.println(number + " is not a perfect number");
    }
}
-----------------------------------------------------------------------------------------
import java.util.Random;
public class OneOfEachStats {
    public static void main (String[] args) {
        int T = Integer.parseInt(args[0]);
        int seed = Integer.parseInt(args[1]);
        Random generator = new Random(seed);
```

```
117         //int number = Integer.parseInt(seed); //amount of families
118         int family2=0;
119         int family3=0;
120         int family4=0;
121         double sum=0;
122         // create family: tempSum - thst sum of children in any family. sum - the sum of
    all the chilsren.
123         for (int i=1; i<=T; i++)
124         {
125         int tempSum=1;
126         boolean boy=false;
127         boolean girl=false;
128         double rnd = generator.nextDouble();
129         //generate the first childern - 2 options. 1=boy, 2=girl. print g/b and add 1 to
    the sum.
130         if(rnd>=0.5 && rnd <1)
131             {
132                 boy=true;
133                 sum=sum+1;
134             }
135         else {
136                 girl=true;
137                 sum=sum+1;
138             }
139         // keep generating until the answer is diffrent from the first child.
140         while(boy!=girl)
141         {
142             rnd = generator.nextDouble();
143             if(rnd>=0.5 && rnd <1)
144             {
145                 sum=sum+1;
146                 tempSum=tempSum+1;
147                 boy = true;
148             }
149             else
150             {
151                 sum=sum+1;
152                 tempSum=tempSum+1;
153                 girl=true;
154             }
155         }
156         // After creating family, check in witch catagory the family is
157         if(tempSum==2)
158             {
159                 family2=family2+1;
160             }
161         else
162             {
163             if(tempSum==3)
164             {
165                 family3=family3+1;
166             }
167             else
168             {
169                 family4=family4+1;
170             }
171         }
172         }
173         int commen = 0;
174         //check witch catagory is most commen
```

```
175         if((Math.max(family2, family3)==family2) && family2>family4)
176         {
177             commen =2;
178         }
179             else
180             {
181             if((Math.max(family3,family4)==family3) && family3>family4)
182             {
183                 commen=3;
184             }
185             else commen=4;
186         }
187         double av = sum/T;
188         System.out.println("Average: " + av + " children to get at least one of each
    gender.");
189         System.out.println("Number of families with 2 children: " + family2);
190         System.out.println("Number of families with 3 children: " + family3);
191         System.out.println("Number of families with 4 or more children: " + family4);
192         System.out.println("The most common number of children is "+commen+".");
193     }
194 }
195
196
```