Sapir Erlich HW2 -

1. Divisors -

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

2. Reverse -

```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 input_str = args[0];
       Integer str_len = input_str.length();
       String reveresed="";
       for ( int i = str_len-1;i >= 0; i--){
           char current_char = input_str.charAt(i);
           reveresed = reveresed + current_char;

       }
       System.out.println(reveresed);
       System.out.println("The middle character is "
       + input_str.charAt((str_len-1)/2));


   }
}
```

3. InOrder -

```
/**
*  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) {
       Integer random_number1 = (int) (Math.random() * 10);
       String random_numbers = "" + random_number1;
       Integer random_number2 = (int) (Math.random() * 10);
       while (random_number1 <= random_number2){
           random_numbers=random_numbers + " " + random_number2;
           random_number1 = random_number2;
           random_number2 = (int) (Math.random() * 10);
       }
       System.out.println(random_numbers);


       }
   }
```

4. DamkaBoard -

```java
/**
*  Gets a command-line argument n (int), and prints an n-by-n damka board.
*/
public class DamkaBoard {
   public static void main(String[] args) {
       Integer number = Integer.parseInt(args[0]);
       int i = 0;
       while (i<number){
           int j = 0 ;
           while (j < number){
               if (i % 2 == 1 ){
               System.out.print(" *");
               }
               else{
                   System.out.print("* ");
               }
               j++;

           }
           System.out.println("");
           i++;

       }
   }
}
```

5. Perfect -

```java
/**
*  Gets a command-line argument (int), and chekcs if the given number is perfect.
*/
public class Perfect {
   public static void main (String[] args) {
       Integer number =Integer.parseInt(args[0]);
       Integer dividers_sum = 1;
       String perfect_output=number+" is a perfect number since "+number+" = 1";
       for (int i = 2 ; i < number ; i++){
           if (number % i == 0){
               dividers_sum = dividers_sum + i;
               perfect_output=perfect_output+" + "+i;

           }
       }

       if ((int)dividers_sum == number){
           System.out.println(perfect_output);
       }
       else{
           System.out.println(number+" is not a perfect number");
       }
   }
}
```

6. 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);
        // starting all counters
        Integer all_childrens  = 0;
        Integer two_child_fam = 0;
        Integer three_child_fam = 0;
        Integer four_plus_child_fam = 0;
        String mode="";
        // a loop that will run T times and generate the one of each stats
        for ( int i = 0 ; i < T ; i++){
            boolean is_boy = false;
            boolean is_girl = false;
            int childrens=0;
            double gender;
            // while there is only boys or only girls keep running
            while (is_boy == false || is_girl == false) {
                gender= generator.nextDouble();
                if (gender<=0.5){
                        is_boy=true;
                        }
                else{
                    is_girl=true;

                }
                childrens++;
            }

            all_childrens = all_childrens + childrens;
            if (childrens==2){
                two_child_fam++;
            }
            else if (childrens==3){
                three_child_fam++;
            }
            else{
                four_plus_child_fam++;
            }
        }
        if (two_child_fam>=three_child_fam && two_child_fam>=four_plus_child_fam ){
            mode="2";
        }
        else if (three_child_fam>=two_child_fam &&
three_child_fam>=four_plus_child_fam ){
            mode="3";
        }
        else{
            mode="4 or more.";
        }
        double avg_childrens = (double)all_childrens / T;
        System.out.println("Average: "+avg_childrens+" children to get at least one
of each gender.");
        System.out.println("Number of families with 2 children: "+two_child_fam);
        System.out.println("Number of families with 3 children: "+three_child_fam);
        System.out.println("Number of families with 4 or more children:
"+four_plus_child_fam);
        System.out.println("The most common number of children is "+mode+".");

    }
}
```