Ejercicio 1

```
public class Maskify {
   public static String maskify(String str) {

     String maskify = "";
          if (srt != null && srt.length() > 0) {
                    if (srt.length() < 4) {
                              return srt;
                    } else {
                              String visible = srt.substring(srt.length() - 4, srt.length());
                              String aux = "";
                              for (int i = 0; i < srt.length() - 4; i++) {
                              aux += '#';
                    }
                    maskify = aux + visible;
          }
  }
return maskify;
   }
}
```

Salida:

```
Maskify.Maskify("4556364607935616"); // should return "############5616"
Maskify.Maskify("64607935616");      // should return "#######5616"
Maskify.Maskify("1");                // should return "1"
Maskify.Maskify("");                 // should return ""
Maskify.Maskify("Skippy");           // should return "##ippy"
Maskify.Maskify("Nanananananananananananananananana Batman!");
// should return "####################################man!"
```

Ejercicio 2:

```
class Kata {
  public static String getMiddle(String word) {
    String getMiddle = "";
          if(word != null && word.length()>0){
                    if(word.length() < 3){
                              return word;
                    }else{
                              if(word.length()%2==0){
                                        getMiddle = word.substring((word.length()/2)-1, (word.length()/2)+1);
                              }else{
                                        getMiddle = word.substring((word.length()-1)/2, (word.length()+1)/2);
                              }
                    }
          }
          return getMiddle;

  }
}
```

Salida:

```
Kata.getMiddle("test") should return "es"

Kata.getMiddle("testing") should return "t"

Kata.getMiddle("middle") should return "dd"

Kata.getMiddle("A") should return "A"
```

Ejercicio 3:

```
public class Fighter {
  public String name;
  public int health, damagePerAttack;
  public Fighter(String name, int health, int damagePerAttack) {
    this.name = name;
    this.health = health;
    this.damagePerAttack = damagePerAttack;
  }
}

public class Kata {
  public static String declareWinner(Fighter fighter1, Fighter fighter2, String firstAttacker) {
                String winner = "none";
                if(fighter1 != null && fighter2 != null && firstAttacker != null){
                        if(firstAttacker.equals(fighter1.name) || firstAttacker.equals(fighter2.name)){
                                while (fighter1.health > 0 && fighter2.health > 0) {
                                        if(firstAttacker.equals(fighter1.name)){
                                                fighter2.health-=fighter1.damagePerAttack;
                                                if(fighter2.health<=0){
                                                        return fighter1.name;
                                                }
                                                fighter1.health-=fighter2.damagePerAttack;
                                                if(fighter1.health<=0){
                                                        return fighter2.name;
                                                }
                                        }else{
                                                fighter1.health-=fighter2.damagePerAttack;
                                                if(fighter1.health<=0){
                                                        return fighter2.name;
                                                }
                                                fighter2.health-=fighter1.damagePerAttack;
                                                if(fighter2.health<=0){
                                                        return fighter1.name;
                                                }
                                        }
                                }
                        }
                }
                return winner;
  }
}
```

Salida:

gana quién tiene la vida <= 0 tras los sucesivos ataques

Pruebas:

```
assertEquals("Lew", Kata.declareWinner(new Fighter("Lew", 10, 2),new Fighter("Harry", 5, 4), "Lew"));
assertEquals("Harry", Kata.declareWinner(new Fighter("Lew", 10, 2),new Fighter("Harry", 5, 4), "Harry"));
assertEquals("Harald", Kata.declareWinner(new Fighter("Harald", 20, 5), new Fighter("Harry", 5, 4), "Harry"));
assertEquals("Harald", Kata.declareWinner(new Fighter("Harald", 20, 5), new Fighter("Harry", 5, 4), "Harald"));
assertEquals("Harald", Kata.declareWinner(new Fighter("Jerry", 30, 3), new Fighter("Harald", 20, 5), "Jerry"));
assertEquals("Harald", Kata.declareWinner(new Fighter("Jerry", 30, 3), new Fighter("Harald", 20, 5), "Harald"));
```

Ejercicio 4:

```
public class Kata
{
  public static void dontGiveMeFive(int start, int end)
  {
      if(start>end){
         int aux = start;
         start = end;
         end = aux;
      }
      for(int i = start ; i <= end ; i++){
         if(i!=5){
         System.out.print(i+", ");
         }
      }
      System.out.println(" -> Result: "+(end-start));
  }
}
```

Entrada/Salida:

```
1,9 -> 1,2,3,4,6,7,8,9 -> Result 8
4,17 -> 4,6,7,8,9,10,11,12,13,14,16,17 -> Result 12
```

Ejercicio 5:

```
public class Codewars {
  public static String oddOrEven (int[] array) {
      String oddOrEven = "error";
      if(array != null && array.length > 0){
         int sum = 0;
         for(int num:array){
            sum+=num;
         }
         if(sum%2==0){
            oddOrEven = "even";
         }else{
            oddOrEven = "odd";
         }
      }
      return oddOrEven;
  }
}
```

Entreda/Salida: cómo es la suma de las entradas…

```
odd_or_even([0])          ==  "even"
odd_or_even([0, 1, 4])    ==  "odd"
odd_or_even([0, -1, -5])  ==  "even"
```

Ejercicio 6:

```
public class Kata {
  public static String highAndLow(String numbers) {
    String highAndLow = "error";
          if(numbers != null && numbers.length()!=5){
             String[] parts = numbers.split(" ");
             int[] num = new int[5];
             for(int i = 0 ; i < num.length ; i++){
                num[i] = Integer.parseInt(parts[i]);
             }
             int high = num[0];
             int low = num[0];
             for(int aux:num){
                if(aux>high)high=aux;
                if(aux<low)low=aux;
             }
          highAndLow = high+" "+low;
          }
       return highAndLow;
  }
}
```

Salida:

```
highAndLow("1 2 3 4 5")  // return "5 1"
highAndLow("1 2 -3 4 5") // return "5 -3"
highAndLow("1 9 3 4 -5") // return "9 -5"
```

Ejercicio 7:

```
public class Solution {

  public static boolean isAscOrder(int[] arr) {
       boolean isAscOrder = false;
       if(arr != null && arr.length > 1){
          for(int i = 0;  i < arr.length -1 ; i++){
             if(arr[i]+1 == arr[i+1]){
                isAscOrder = true;
             }else isAscOrder = false;
          }
       }
       return isAscOrder;
  }

}
```

Salida:

```
isAscOrder(new int[]{1,2,4,7,19}) == true
isAscOrder(new int[]{1,2,3,4,5}) == true
isAscOrder(new int[]{1,6,10,18,2,4,20}) == false
isAscOrder(new int[]{9,8,7,6,5,4,3,2,1}) == false // numbers are in DESCENDING
order
```

Ejercicio 8:

```
public class WhichAreIn {

        public static String[] inArray(String[] array1, String[] array2) {
                        String[] inArray = {""};
      int nCoincidencias = 0;
      if(array1 != null && array2 != null && array1.length > 0 && array2.length > 0){
         for(int i = 0 ; i < array1.length ; i++){
            for(int j = 0; j < array2.length ; j++){
              if(array2[j].contains(array1[i])){
                 nCoincidencias++;
                 j=array2.length;
              }
            }
         }
         if(nCoincidencias != 0) inArray = new String[nCoincidencias];
         for(int i = 0, x = 0 ; i < array1.length ; i++){
            for(int j = 0; j < array2.length ; j++){
              if(array2[j].contains(array1[i])){
                 inArray[x] = array1[i];
                 x++;
                 j=array2.length;
              }
            }
         }
      }
      return inArray;
         }
}
```

Entrada/Salida:

#Example 1: `a1 = ["arp", "live", "strong"]`

`a2 = ["lively", "alive", "harp", "sharp", "armstrong"]`

returns `["arp", "live", "strong"]`

#Example 2: `a1 = ["tarp", "mice", "bull"]`

`a2 = ["lively", "alive", "harp", "sharp", "armstrong"]`

returns `[]`

Ejercicio 9:

```
public class EnoughIsEnough {

        public static int[] deleteNth(int[] elements, int maxOccurrences) {
        //Está feisimo y sin comentar, me vas a matar, pero ahora mismo no me da la cabeza ni para hablar bien :'(
                        int[] deleteNth = null;
      int newTam = 0;
      int j = 0;
      int aux = -999999999;

      if (elements != null && elements.length > 0 && maxOccurrences > 0) {
         for (int i = 0; i < elements.length; i++) {
            j = 0;
            for (int x = 0; x < elements.length; x++) {
               if (elements[i] == elements[x]) {
                  j++;
               }
            }
            if(j <= maxOccurrences){
               newTam++;
            }else if(elements[i] != aux){
               newTam+=maxOccurrences;
               aux = elements[i];
            }
         }
         aux = -999999999;
         int contador = 0;
         deleteNth = new int[newTam];
         for (int i = 0; i < elements.length; i++) {
            j = 0;
            for (int x = 0; x < elements.length; x++) {
               if (elements[i] == elements[x]) {
                  j++;
               }
            }
            if(j <= maxOccurrences){
               deleteNth[contador]=elements[i];
               contador++;
            }else if(elements[i] != aux){
               for(int q = 0; q < maxOccurrences ; q++,contador++){
                  deleteNth[contador] = elements[i];
               }
               aux = elements[i];
            }
         }
      }
      return deleteNth;
         }

}
```

Entrada/Salida:

```
EnoughIsEnough.deleteNth(new int[] {20,37,20,21}, 1) // return [20,37,21]
EnoughIsEnough.deleteNth(new int[] {1,1,3,3,7,2,2,2,2}, 3) // return [1, 1, 3,
3, 7, 2, 2, 2]
```

Ejercicio 10:

```
public class DRoot {
  public static int digital_root(int n) {
        int cifra = 0;
        int suma = 0;
        boolean entra = true;

        while (entra) {
           while (n != 0) {
              cifra = n % 10;
              suma = suma + cifra;
              n = n / 10;
           }
           if (suma > 9) {
              n = suma;
              suma = 0;
              cifra = 0;
           } else
              entra = false;
        }
        return suma;
  }
}
```

Entrada/Salida:

```
16  -->  1 + 6 = 7
942  -->  9 + 4 + 2 = 15  -->  1 + 5 = 6
132189  -->  1 + 3 + 2 + 1 + 8 + 9 = 24  -->  2 + 4 = 6
493193  -->  4 + 9 + 3 + 1 + 9 + 3 = 29  -->  2 + 9 = 11  -->  1 + 1 = 2
```