public void needHelpWithProject(String instructor) {

    switch (instructor) {

    case "TA":
        System.out.println("Welcome to send e-mails");
        System.out.println("Do come to Tuesday office hours");
        System.out.println("Do ask questions during discussions as well");

    case "SI":
        System.out.println("No");

    case "Prof":
        System.out.println("No technical e-mails !");
        System.out.println("Do not ask questions on projects after class !");
        String tip1="1- Come prepare with very specific questions";
        String tip2="2- I will not look at your code";
        String tip3="3- Any other matter/concerns welcome";
        System.out.println("During my office hours: "+tip1+tip2+tip3);

    }
}
Why Sorting?

- **Organize large database.** Examples:
  - arrange customer names in alphabetical order, or by ZIP code, etc.
  - home sales by prices, by days in the market, or by square feet, etc.
  - cities by population, average household income, land size, etc.
  - Class notes of ECE242 by topics, and so on....

- Sorting data is often a **primarily step to searching** it (using binary search for example)

- Sorting has been the subject of extensive research in CS

- We will here first look at 3 simple sorting algorithms:
  - 1- Bubble sort
  - 2- Selection sort
  - 3- Insertion sort

Those are easier to understand/analyze at first, and they could also be better than more sophisticated algorithms in some situations
Introduction

- How would you sort this team from shortest to tallest player?

- As a human being, we can look at all the player at once and immediately pick the shortest one.

- A computer program isn't able to see the big picture. It must rely on performing basic steps (follow simple rules):
  - Compare two players at a time
  - Swap two players
  - Remove or Insert one player
  - Move/Shift a player
Introduction

- Each sorting algorithm handles the details of the basic steps differently.
- The end result should (hopefully:-) be the same.

- To understand the relative differences between algorithms, we need to learn more about:
  - their $O$ complexity
  - their practical efficiency in particular situations
Simple Sorting Algorithms

1- Bubble Sort
2- Selection Sort
3- Insertion Sort
1- Bubble Sort - overview

- Example: sort in ascending order

Main Step - sort N items:
- Start from the left
- Compare neighbors and swap them if not in order
- Move one position right and go back to previous step

Cost:
N-1 comparisons and 0 to N-1 swaps

Go Back to Main Step but sort for the first N-1 items left
1- Bubble Sort- Examples

- For fun: https://www.youtube.com/watch?v=lyZQPjUT5B4

- To do: Test Java applet BubbleSort.html
The basic bubbleSort method is only few lines long (we use array of primitive type int rather than object for simplicity)

```java
public void bubbleSort()
{
    int in,out;
    int temp;
    for(out=N-1; out>0; out--) // outer loop (backward)
        for(in=0; in<out; in++) // inner loop (forward)
            if(array[in] > array[in+1]) // out of order? swap them
            {
                temp = array[in];
                array[in] = array[in+1];
                array[in+1] = temp;
            }
} // end bubbleSort()
```

Complexity analysis: Two loops
- Sum steps in inner loop: \((N-1)+(N-2)+...+1=N^*(N-1)/2\) \(\rightarrow O(N^2)\)
- Number of comparisons is \(O(N^2)\)
- Number of swaps is smaller than comparisons but still \(O(N^2)\)
- BubbleSort is very slow (in particular it has too many swaps)
Simple Sorting Algorithms

1- Bubble Sort
2- Selection Sort
3- Insertion Sort
2- Selection Sort - overview

- Example: sort in ascending order

Main Step - sort N items:
- Find the smallest item
- Swap it with the item at first position

Go Back to Main Step but sort for the last N-1 items left

Cost: N-1 comparisons and 1 swap

Results after successive steps of Selection Sort
2- Selection Sort- Examples

- For fun: https://www.youtube.com/watch?v=Ns4TPTC8whw

- To do: Test Java applet SelectSort.html
2- Selection Sort - Algorithm

- The basic selectionSort method is also few lines long

```java
public void selectionSort()
{
    int in, out, min;
    int temp;  // temp variable
    for(out=0; out<N-1; out++)  // outer loop
    {
        // find the minimum item between [out+1,N-1]
        min=out;  // initialize minimum index
        for(in=out+1; in<N; in++)  // inner loop
            if(array[in] < array[min]) min=in;  // update minimum index
        // swap item "array[min]" with item "array[out]"
        temp = array[out];
        array[out] = array[min];
        array[min] = temp
    }  // end outer loop
}  // end selectionSort()
```

- Complexity analysis:
  - Same number of comparisons than bubbleSort (two loops) $O(N^2)$
  - Number of swaps depends only on outer loop, it is then $O(N)$
  - Selection Sort is $O(N^2)$ but it is expected to be faster than BubbleSort