Important

There are a few guidelines you must follow in this homework. If you fail to follow any of the following guidelines you will receive a **0** for the entire assignment.

1. All submitted code must compile under **JDK 7**. This includes unused code, don't submit extra files that don't compile. (Java is backwards compatabile so if it compiles under JDK 6 it *should* compile under JDK 7)
2. Don't include any package declarations in your classes.
3. Don't change any *existing* class headers or method signatures. (It is fine to add extra methods and classes)
4. Don't import anything that would trivialize the assignment. (e.g. don't import `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
5. You must submit your source code, the `.java` files, not the compiled `.class` files.

After you submit your files redownload them and run them to make sure they are what you intended to submit. We are not responsible if you submit the wrong files.

Instructions

Complete the missions below as best you can. Make sure you read the dialogs and the javadocs in each class so that you thoroughly understand the task at hand. Feel free to jump from mission to mission. (For example, basic training may be pretty hard, so you can do that one last if you want).

Prelude:

General Davis: Good morning, private. You are currently enlisted in one of the most honorable coding forces in the nation! My name is General Davis. After basic training, you will report directly to me. I have a few important missions for you. This is Sgt. Reddaway. He will be your mentor until you get up to speed.

Sgt. Reddaway: We have a lot of work ahead of us.

General Davis: That's what I like to hear! Here in Batallion #CC1332 nobody gets to sleep! Now get off to basic training! Move, move, move!

Basic Training: `BasicTraining.java`

Sgt. Reddaway: Here in battalion #CC1332, you either sink or swim. So, your only test for basic training is to code Quicksort. Muhahaha.

Mission 1: Sorting Soldiers `MissionOne.java`

General Davis: You may not have noticed, but as every soldier completes basic training, we evaluate them. That way we can create squadrons that are all about the same strength. Back home, the President has been pressuring me to send him a file of the results ASAP. Of course, he wants them listed in descending order with the best soldier first. Problem is that our computer is just one of those field laptops and we do not have enough RAM to store much extra data in memory so you will need to write the sort in-place.

Sgt. Reddaway: Hmm . . . so it looks like we wont be able to use any extra arrays to do the sorting. One thing to do would be to find the best soldier in the array and move him to the front. Problem is, you will have to ignore that soldier the next time you are finding the best soldier.

Mission 2: Selecting Targets `MissionTwo.java`

General Davis: The enemy is attacking! There are so many targets! Which one do we shoot first?

Private Hull: The one that is more important! Duh.

General Davis: But we can't have everyone shooting at the same thing! That would be a waste. Private! I need you to solve this problem! I'll give you targets as the battle progresses, and you need to tell me which ones are the highest priority without delay. Get moving!

Sgt. Reddaway: It sounds like the General will be both adding stuff to your list and looking up targets from it. It also sounds like it is more important for him to retrieve targets faster than adding them. In order for him to retrieve targets quickly, we should probably have the list sorted before he tries to retrieve from it. This means we should sort the list whenever he adds a new target. Or better yet, add each new target where it would go to maintain the list's sorted order.

General Davis: I'll also need to be able to remove targets once they are destroyed!

Sgt. Reddaway: That shouldn't be hard.

Mission 3: Award Ceremony `MissionThree.java`

General Davis: Private, your coding ability was admirable the past few battles. Im considering you for a promotion!

Sgt. Reddaway: And what about the person who trained him?

General Davis: You still haven't written that program I asked for!

Sgt. Reddaway: Oh yea . . .

General Davis: Anyway, after a day of fighting, Colonel Robert, the other officers, and I like to wind down at the bars. Of course all we talk about is how many medals we have won – which is a lot. I have the most of course.

Colonel Robert: But Kyle, yours aren't better than mine! Anyway, all of the officers have decided to play a game to see who the most prestigious is. Each match is only between two officers. First, we merge their medals together to make a single, sorted list of medals. Then we give points to each officer depending on where their medals are in the list. The least prestigious medal earns 1 point, the second least prestigious earns 2 points, the third least prestigious earns 3 points, and so on. For example, if the sorted array of medals from least prestigious to most prestigious looked like this (D for Davis and R for Robert) `DDDRDRRDRD`

General Davis would get $1 + 2 + 3 + 5 + 8 + 10 = 29$ points.

Colonel Robert would get $4 + 6 + 7 + 9 = 26$ points.

In this case, General Davis would win by 3 points.

General Davis: I need you to write a program that will speed up the game for us. I will give you two arrays of medals, and I need you to return the difference in score between the first and second person. To help you out, I will make sure that the arrays of medals that I give you will already be sorted.

Mission Impossible: Optimal Attack Zone `MissionImpossible.java`

This is for 20 points of extra credit. Attempting this, or ignoring this section will not hurt your grade.

General Davis: We have a problem. The enemy has set up defensive watch towers all over the field we were going to mount an assault from. Now we do not know where to drop our troops. These towers are equipped with very powerful lasers that can shoot any distance. Lucky for us they are only equipped to fire horizontally and vertically.

Sgt. Reddaway: But what can we do about it, it sounds like these lasers are going to tear through our troops?

General Davis: That's why we are going to be strategic. We will land as many troops as we can in the largest rectangle that the lasers can't hit. Unfortunately I'm not sure how to figure this problem out, the enemy has a large amount of territory. Our spies found out the coordinates of all the towers, but they aren't in any particularly usefull order. They also notice that no two towers are able to hit each other.

Sgt. Reddaway: Well how much territory do we have to look over?

General Davis: I couldn't tell how big it was, the spies never reached the end of the field. By our estimates no tower is more than 2 billion meters away from the center of the field in either the horizontal or vertical direction.

Colonel Robert: Well General does the size of the field really matter? I would like to know how many towers we're looking at avoiding to land these troops.

General Davis: Good point. I'm not sure if it's important. My sources tell me that there are no more than 10 million towers. And I need to be able to solve this quickly in case the state of the field changes. I can't afford to have a solution that takes more than a few seconds. Private! Are you up for the task? I'm looking for a solution that's $O(n)$ that will tell me the largest continguous piece of land I can put my troops on. (The towers are 1 square meter in size and fire lasers 1 meter in width veritcally and horizontaly. Each troop fits in a 1 square meter section of field. Return the number of troops I can land in the field without lasers splitting them up)

Sgt. Reddaway: This is a tough problem. It's beyond my skills, I don't think I will be much help. You're going to need to figure this one out on your own. *(Don't ask the TA's for the answer, this is extra credit, you will need to think about it! – remember this is part of a sorting homework, that should be a hint at what sort of algorithms you need to use)*

Deliverables

You must submit all of the following files.

1. `BasicTraining.java`
2. `MissionOne.java`
3. `MissionTwo.java`
4. `MissionThree.java`
5. `MissionImpossible.java`

You may attach them each individually, or submit them in a zip archive.