1DV512 - Lab assignment 1
Student: Marah Awad

In this assignment, we have to simulate contiguous memory allocation by different strategies. I started my application by creating a class “Block.java” which simulates free blocks in the memory and has a start and end address in it, and I added setters and getters to its attributes.
After that I created a file “AllocatedBlock.java” which will simulate allocated blocks, I make it extend the parent class “Block” and differ only with Id for allocated blocks. And then I started with “Memory.java” simulating memory and having a list of free blocks and one for allocated blocks. And the memory constructor has the size needed to create the memory, so when memory is created, it consists of only one free block with start address = 0, and end address = size -1. This class contains getters and setters methods, in addition to adding and removing for arrays in the class. I created three additional classes that simulate three strategies we need to simulate. My idea is that working on memory depends on the strategy we have. So the parent class is “Memory” and the three other classes are “FirstFitMemory”, “BestFitStrategy” and “WorstFitStrategy”. They differ on Allocate-method and definitely memory sorting. I started with firstFit memory, its allocate-method first checks if the size of the block attempted to allocate is bigger than the whole memory a message throws to OutOfMemoryError collector. Looping into free blocks and if the size of the block I want to allocate fits in a free block. I create it with the start and end addresses then add it to the allocated blocks list -more details in code comments-. My code for strategy allocation depends firsthand on how I sort the free blocks. For example, for the best fit strategy I sort free blocks from the smallest to the bigger sequentially. For the First Fit strategy I sort them by starting address, so the lowest starting address to the highest starting address sequentially. And lastly, for the Worst Fit Strategy, the sorting is from the bigger block to the smallest sequentially. So when I want to allocate I sort free blocks first and then try to allocate depending on the size. In my work, I coded first the First Fit Strategy and created the “Reader.java” class to read the file and to try and debug my allocation to check it all the time by debugging and printing in the terminal.
For deallocating I create the deallocate-method in the parent class “memory” using the allocated block Id have to be deallocated, and it's the same method for all three different strategies. Then I added the body to the two other strategies and tried to run them.
I added the fragmentation method in the memory class and methods needed to calculate fragmentation such as “getFreeBlocksSize()” to get the sum of all free blocks in memory and method to get the largest block and its size.
After that, I added to reader class methods to create and write my output from my code. I tried to create different input scenarios in the scenario.in file to check if my strategy is working right or not. And for different input I tried with, the best fit strategy gave always the smallest fragmentation. At the end I created compact()-method in the memory -parent class- To compact the memory I move the start address to the first allocated block of the array to 0 and make no free blocks between allocated blocks then delete all free blocks from the free blocks array and add one free block that starts from the end of the last allocated block.