Τα αρχεία του φακέλου έχουν το καθένα την εξής λειτουργία:

- Node.java: το αρχείο με την υλοποίση των κόμβων της λίστας
- Processed_jobs.java: η υλοποίηση της λίστας
- Processor.java: ο ΑΤΔ δεδομένων για την αναπαράσταση ενός επεξεργαστή
- MaxPQ.java: η ουρά προτεραιότητας
- ProcessorComparator.java: η υλοποίηση για τα compare της MaxPQ
- readFile.java: η υλοποίηση του διαβάσματος ενός αρχείου
- Greedy.java: ο Αλγόριθμος 1
- Sort.java: το αρχείο ταξινόμησης
- writeFiles.java: το γράψιμο αρχείων (προαιρετικό)
- Comparisons.java: το μέρος Δ

Αλγόριθμος 1:

Για την υλοποίση του αλγορίθμου 1 εργαστήκαμε ως εξής: στο αρχείο readFile.java έχουμε την υλοποίηση του διαβάσματος ενος αρχείου .txt. Έτσι στο Greedy.java φτιάχνουμε ένα τέτοιο αντικέιμενο και με την loadFile(String path) φορτώνουμε το αρχείο. Έπειτα δημιουργούμε μια λίστα κενή στην οποία με τη μέθοδο setList(int jobs) περνάμε στη λίστα τις διεργασίες που έχουμε διαβάσει από το αρχείο. Μετά δημιουργούμε μια ουρά προτεραιότητας στην οποία βάζουμε τόσους επεξεργαστές όσους μας λέει το αρχείο. Μετά σε ένα loop δημιουργούσαμε έναν επεξεργαστή και σε αυτόν περνούσαμε κάθε φορά τον επεξεργαστή με την ελάχιστη προτεραιότητα από την MaxPQ. Αυτό το στοιχείο μετά το περνούσαμε πάλι στην MaxPQ. Έτσι μετά το τέλος της επανάληψης σε κάθε επεξεργαστή περνούσε η διεργασία που του αναλογούσε (σύμφωνα με το παράδειγμα του Αλγορίθμου 1). Τέλος τυπώθηκαν οι επεξεργαστές σε αύξουσα σειρά, και πάλι με την χρήση του getMin από την ουρά βρέθηκε το Makespan.

Ταξινόμηση:

Για την ταξινόμηση έγινε χρήση του αλγορίθμου Mergesort σε πίνακα. Επειδή μέχρι τώρα τα δεδομένα αποθηκεύονταν σε λίστα, για την υλοποίηση του Αλγορίθμου 2 περάσαμε τα στοιχεία της λίστας σε έναν πίνακα με το αντίστοιχο μέγεθος. Προτιμήθηκε η χρήση πινάκων γιατί εμφανίζει περισσότερες ευκολίες σε σχέση με τις λίστες όσων αφορά την ταξινόμησή τους και επίσης η άσκηση δεν σε οδηγεί αναγκαστικά σε δυναμική αποθήκευση των δεδομένων (τότε αναγκαστικά θα χρησιμοποιούσαμε λίστες). Η υλοποίηση της ταξινόμησης εγίνε με βάση το πρότυπο των διαφανειών και του βιβλίου του μαθήματος.

Μέρος Δ:

Στον παρακάτω πίνακα παρουσίαζονται τα Makespan απο την εκτέλεση του προγράμματος στο μέρος Δ. Εκτελέσαμε το πρόγραμμα 4 φορές για να πάρουμε τα συγκεκριμένα στοιχεία. Στην πρώτη δοκιμή είναι η έξοδος του προγράμματος με βάση τα αρχεία που υπάρχουν στον φάκελο data.

N	100		500		1000	
Greedy	1st	1039	1st	1922	1st	2700
	2nd	1053	2nd	1913	2nd	2720
	3rd	1049	3rd	1916	3rd	2717
	4th	1041	4th	1940	4th	2742
Sort	1st	1014	1st	1900	1st	2687
	2nd	1024	2nd	1897	2nd	2707
	3rd	1012	3rd	1900	3rd	2707
	4th	1010	4th	1923	4th	2730

Παρατηρούμε το αναμενόμενο, ότι δηλαδή με την ταξινόμηση των διεργασιών, το makespan μειώνεται και έτσι εκτελείται πιο αποτελεσματικά το όλο πρόγραμμα. Ωστόσο η απόκλιση στο makespan του greedy με αυτό του sort δεν είναι πολύ μεγάλη. Τέλος μια άλλη παρατήρηση είναι το γεγονός ότι όσο περισσότερες είναι οι διεργασίες τόσο μικρότερη είναι η απόκλιση στο makespan των 2 αλγορίθμων. Αυτό βέβαια δεν μπορούμε να το πούμε με σιγουριά καθώς τα δεδομένα μας βασίζονται σε τυχαίους αριθμούς.