SchedulingAlgorithms.java:-

```java
import java.util.*;

public class SchedulingAlgorithms {

    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n==============================");
            System.out.println(" Select Scheduling Algorithm ");
            System.out.println("==============================");
            System.out.println("1. First Come First Serve (FCFS)");
            System.out.println("2. Shortest Job First (SJF - Preemptive)");
            System.out.println("3. Priority Scheduling (Non-Preemptive)");
            System.out.println("4. Round Robin (RR)");
            System.out.println("5. Exit");
            System.out.print("Enter choice: ");

            int choice = sc.nextInt();
            switch(choice) {
                case 1: fcfs(); break;
                case 2: sjf(); break;
                case 3: priority(); break;
                case 4: rr(); break;
                case 5:
                    System.out.println("Exiting... Thank you SWARAJ!");
                    return; // End program
                default:
                    System.out.println("Invalid choice! Try again.");
            }
        }
    }

    // ==================== FCFS ====================
    static void fcfs() {
        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();
        int pid[] = new int[n], at[] = new int[n], bt[] = new int[n], ct[] = new
int[n], tat[] = new int[n], wt[] = new int[n];

        for (int i = 0; i < n; i++) {
            pid[i] = i+1;
            System.out.print("Enter AT and BT for P" + pid[i] + ": ");
            at[i] = sc.nextInt();
            bt[i] = sc.nextInt();
        }

        // Sort by Arrival Time
        for(int i=0;i<n-1;i++){
            for(int j=0;j<n-i-1;j++){
                if(at[j] > at[j+1]){
                    int temp=at[j]; at[j]=at[j+1]; at[j+1]=temp;
                    temp=bt[j]; bt[j]=bt[j+1]; bt[j+1]=temp;
                    temp=pid[j]; pid[j]=pid[j+1]; pid[j+1]=temp;
                }
            }
        }

        int time=0; double avgWT=0;
        for(int i=0;i<n;i++){
            if(time<at[i]) time=at[i];
            ct[i]=time+bt[i];
            time=ct[i];
            tat[i]=ct[i]-at[i];
            wt[i]=tat[i]-bt[i];
            avgWT+=wt[i];
        }

        System.out.println("\n--- First Come First Serve ---");
        printResult(pid, wt, avgWT/n);
    }

    // ==================== SJF Preemptive ====================
    static void sjf() {
        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();
        int pid[] = new int[n], at[] = new int[n], bt[] = new int[n], rt[] = new
int[n], ct[] = new int[n], tat[] = new int[n], wt[] = new int[n];
        boolean completed[] = new boolean[n];

        for (int i=0;i<n;i++) {
            pid[i]=i+1;
            System.out.print("Enter AT and BT for P" + pid[i] + ": ");
            at[i]=sc.nextInt();
            bt[i]=sc.nextInt();
            rt[i]=bt[i];
        }

        int finished=0, time=0; double avgWT=0;
        while(finished<n){
            int idx=-1, minRT=Integer.MAX_VALUE;
            for(int i=0;i<n;i++){
                if(!completed[i] && at[i]<=time && rt[i]<minRT && rt[i]>0){
                    minRT=rt[i]; idx=i;
                }
            }
            if(idx==-1){ time++; continue; }
            rt[idx]--; time++;
            if(rt[idx]==0){
                completed[idx]=true;
                finished++;
                ct[idx]=time;
                tat[idx]=ct[idx]-at[idx];
                wt[idx]=tat[idx]-bt[idx];
                avgWT+=wt[idx];
            }
        }

        System.out.println("\n--- Shortest Job First (Preemptive) ---");
        printResult(pid, wt, avgWT/n);
    }

    // ==================== Priority Scheduling ====================
    static void priority() {
        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();
        int pid[] = new int[n], at[] = new int[n], bt[] = new int[n], pr[] = new
int[n], ct[] = new int[n], tat[] = new int[n], wt[] = new int[n];
        boolean done[] = new boolean[n];

        for(int i=0;i<n;i++){
            pid[i]=i+1;
            System.out.print("Enter AT, BT, Priority for P" + pid[i] + ": ");
            at[i]=sc.nextInt(); bt[i]=sc.nextInt(); pr[i]=sc.nextInt();
        }
        int time=0, completed=0; double avgWT=0;
        while(completed<n){
            int idx=-1, bestPr=Integer.MAX_VALUE;
            for(int i=0;i<n;i++){
                if(!done[i] && at[i]<=time && pr[i]<bestPr){
                    bestPr=pr[i]; idx=i;
                }
            }
            if(idx==-1){ time++; continue; }
            time+=bt[idx];
            ct[idx]=time;
            tat[idx]=ct[idx]-at[idx];
            wt[idx]=tat[idx]-bt[idx];
            done[idx]=true; completed++;
            avgWT+=wt[idx];
        }

        System.out.println("\n--- Priority Scheduling (Non-Preemptive) ---");
        printResult(pid, wt, avgWT/n);
    }

    // ==================== Round Robin ====================
    static void rr() {
        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();
        int pid[] = new int[n], at[] = new int[n], bt[] = new int[n], rt[] = new
int[n], ct[] = new int[n], tat[] = new int[n], wt[] = new int[n];

        for(int i=0;i<n;i++){
            pid[i]=i+1;
            System.out.print("Enter AT and BT for P" + pid[i] + ": ");
            at[i]=sc.nextInt(); bt[i]=sc.nextInt(); rt[i]=bt[i];
        }

        System.out.print("Enter Time Quantum: ");
        int q=sc.nextInt();

        Queue<Integer> qList=new LinkedList<>();
        boolean added[]=new boolean[n];
        int time=0, completed=0; double avgWT=0;

        while(completed<n){
            for(int i=0;i<n;i++){
                if(!added[i] && at[i]<=time){
                    qList.add(i); added[i]=true;
                }
            }
            if(qList.isEmpty()){ time++; continue; }
            int idx=qList.poll();
            int exec=Math.min(rt[idx],q);
            rt[idx]-=exec; time+=exec;
            for(int i=0;i<n;i++){
                if(!added[i] && at[i]<=time){
                    qList.add(i); added[i]=true;
                }
            }
            if(rt[idx]>0){ qList.add(idx); }
            else{
                completed++; ct[idx]=time; tat[idx]=ct[idx]-at[idx];
wt[idx]=tat[idx]-bt[idx]; avgWT+=wt[idx];
            }
        }

        System.out.println("\n--- Round Robin Scheduling ---");
        printResult(pid, wt, avgWT/n);
    }

    // ==================== Helper Function ====================
    static void printResult(int pid[], int wt[], double avgWT){
        System.out.println("Process ID\tWaiting Time");
        for(int i=0;i<pid.length;i++){
            System.out.println("P"+pid[i]+"\t\t"+wt[i]);
        }
        System.out.println("Average Waiting Time = " + avgWT);
    }
}
```

OUTPUT:-

```
swaraj@swaraj-VirtualBox:~/LP-1$ javac SchedulingAlgorithms.java
swaraj@swaraj-VirtualBox:~/LP-1$ java SchedulingAlgorithms

==============================
 Select Scheduling Algorithm
==============================
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF - Preemptive)
3. Priority Scheduling (Non-Preemptive)
4. Round Robin (RR)
5. Exit
Enter choice: 1
Enter number of processes: 4
Enter AT and BT for P1: 0 5
Enter AT and BT for P2: 1 3
Enter AT and BT for P3: 2 8
Enter AT and BT for P4: 3 6

--- First Come First Serve ---
Process ID  Waiting Time
P1          0
P2          4
P3          6
P4          13
Average Waiting Time = 5.75

==============================
 Select Scheduling Algorithm
==============================
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF - Preemptive)
3. Priority Scheduling (Non-Preemptive)
4. Round Robin (RR)
5. Exit
Enter choice: 2
Enter number of processes: 2
Enter AT and BT for P1: 0 8
Enter AT and BT for P2: 1 4

--- Shortest Job First (Preemptive) ---
Process ID  Waiting Time
P1          4
P2          0
Average Waiting Time = 2.0

==============================
 Select Scheduling Algorithm
```

```
==============================
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF - Preemptive)
3. Priority Scheduling (Non-Preemptive)
4. Round Robin (RR)
5. Exit
Enter choice: 3
Enter number of processes: 4
Enter AT, BT, Priority for P1: 0 8 2
Enter AT, BT, Priority for P2: 1 4 1
Enter AT, BT, Priority for P3: 2 9 3
Enter AT, BT, Priority for P4: 3 5 2

--- Priority Scheduling (Non-Preemptive) ---
Process ID  Waiting Time
P1          0
P2          7
P3          15
P4          9
Average Waiting Time = 7.75

==============================
 Select Scheduling Algorithm
==============================
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF - Preemptive)
3. Priority Scheduling (Non-Preemptive)
4. Round Robin (RR)
5. Exit
Enter choice: 4
Enter number of processes: 4
Enter AT and BT for P1: 0 8
Enter AT and BT for P2: 1 4
Enter AT and BT for P3: 2 9
Enter AT and BT for P4: 3 6
Enter Time Quantum: 3

--- Round Robin Scheduling ---
Process ID  Waiting Time
P1          16
P2          11
P3          16
P4          13
Average Waiting Time = 14.0

==============================
 Select Scheduling Algorithm
==============================
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF - Preemptive)
3. Priority Scheduling (Non-Preemptive)
4. Round Robin (RR)
5. Exit
Enter choice: 5
Exiting... Thank you SWARAJ!
```