```java
public class Main {

    public static void main(String[] args) throws InterruptedException {
        FCFS fcfs = new FCFS();
        SJF sjf = new SJF();
        RR rr = new RR( timeSlice: 13);

        fcfs.execute();
        sjf.execute();
        rr.execute();

        System.out.println();
        System.out.println("The Scheduling Algorithm Evaluation end" );
    }
}
```

```java
public class Job implements Runnable, Comparable<Job> {

    String pid;
    int num;
    long startTime;
    int tmp = 0;
    int i;

    @Override
    public void run() {
        this.i = num;
        this.startTime = System.currentTimeMillis();
        for (; i >= 0; i--) {
            tmp++;
        }
    }

    public Job(int pid, int num) {
        this.pid = "Job " + pid;
        this.num = num;
    }

    @Override
    public int compareTo(Job o) {
        return this.num - o.num;
    }
}
```

```java
import ...

public class FCFS {

    private ExecutorService executor = Executors.newSingleThreadExecutor();

    public void execute() throws InterruptedException {
        double sum = 0;
        LinkedList<Job> list = new LinkedList<>();
        long[] waitingTimes = new long[4];

        for (int i = 4; i >= 1; i--) {
            list.add(new Job(i,  num: i * 50000000));
        }

        long now = System.currentTimeMillis();

        for (Job job : list) {
            executor.submit(job);
        }

        executor.shutdown();
        executor.awaitTermination( timeout: 10, TimeUnit.SECONDS);

        for (int i = 0; i < 4; i++) {
            waitingTimes[i] = list.get(i).startTime - now;
        }

        long error = waitingTimes[0];

        for (int i = 0; i < 4; i++) {
            waitingTimes[i] -= error;
            sum += waitingTimes[i];
        }

        System.out.println("============ FCFS ============");
        System.out.println("----------- Gantt chart -----------");
        for (int i = 0; i < 4; i++) {
            System.out.println(list.get(i).pid + " starting time: " + waitingTimes[i]);
        }
        System.out.println("-----------------------------");

        for (int i = 0; i < 4; i++) {
            System.out.println(list.get(i).pid + " waiting time: " + waitingTimes[i]);
        }
        System.out.println("---> average waiting time: " + sum / 4);
    }
}
```

```java
import ...

public class SJF {

    private ExecutorService executor = Executors.newSingleThreadExecutor();

    public void execute() throws InterruptedException {
        double sum = 0;
        LinkedList<Job> list = new LinkedList<>();
        long[] waitingTimes = new long[4];

        for (int i = 4; i >= 1; i--) {
            list.add(new Job(i,  num: i * 50000000));
        }

        Collections.sort(list);

        long now = System.currentTimeMillis();

        for (Job job : list) {
            executor.submit(job);
        }

        executor.shutdown();
        executor.awaitTermination( timeout: 10, TimeUnit.SECONDS);

        for (int i = 0; i < 4; i++) {
            waitingTimes[i] = list.get(i).startTime - now;
        }

        long error = waitingTimes[0];

        for (int i = 0; i < 4; i++) {
            waitingTimes[i] -= error;
            sum += waitingTimes[i];
        }

        System.out.println("============ SJF ============");
        System.out.println("---------- Gantt chart ----------");
        for (int i = 0; i < 4; i++) {
            System.out.println(list.get(i).pid + " starting time: " + waitingTimes[i]);
        }
        System.out.println("-----------------------------");

        for (int i = 0; i < 4; i++) {
            System.out.println(list.get(i).pid + " waiting time: " + waitingTimes[i]);
        }
        System.out.println("---> average waiting time: " + sum / 4);
    }
}
```

```java
import ...

public class RR {

    private final double timeSlice;

    public RR(double timeSlice) {
        this.timeSlice = timeSlice;
    }

    public void execute() throws InterruptedException {
        double sum = 0;
        LinkedList<Job> list = new LinkedList<>();
        ArrayList<ArrayList<Long>> result = new ArrayList<>();
        for (int i = 0; i <= 4; i++) {
            result.add(new ArrayList<>());
        }

        for (int i = 4; i >= 1; i--) {
            list.add(new Job(i,  num: i * 50000000));
        }

        System.out.println("============ RR ============");
        System.out.println("---------- Gantt chart ----------");

        long now = System.currentTimeMillis();

        while (!list.isEmpty()) {
            ExecutorService executor = Executors.newSingleThreadExecutor();
            int tmp = 0;
            Job job = list.poll();
            long tmpNow = System.currentTimeMillis();

            executor.execute(job);
            while (System.currentTimeMillis() - tmpNow < timeSlice) {
                tmp++;
            }
            executor.shutdown();
            if (job.i > 0) {
                list.add(new Job(Character.getNumericValue(job.pid.charAt(4)), job.i));
            }
            result.get(Character.getNumericValue(job.pid.charAt(4))).add(job.startTime - now);
            System.out.println(job.pid + " starting time: " + (job.startTime - now));
        }

        Long error = result.get(4).get(0);
        for (ArrayList<Long> longs : result) {
            longs.stream().map((i) -> i - error);
        }


        System.out.println("-----------------------------");

        for (int i = 3; i >= 0; i--) {
            long tmp = 0;
            ArrayList<Long> longs = result.get(i + 1);
            if (!longs.isEmpty()) {
                tmp += longs.get(0);
            }
            int size = longs.size();
            for (int j = 1; j < size; j++) {
                tmp += longs.get(j) - longs.get(j - 1) - timeSlice;
            }
            System.out.println("Job " + (i + 1) + " waiting time: " + tmp);
            sum += tmp;
        }
        System.out.println("---> average waiting time: " + sum / 4);
    }
}
```

```
hyungwook@hyungwocBookPro:~/Desktop/학교강의/운영체제/운체과제2/운체과제/out/pr...
 12:51:40 PM  -44  hyungwook  ~/Desktop/학 ㄱ 교강 ㅇ 영운 ㅇ 체제 ㅈ 제/운 ㅊ
제과 ㅈ 제2/운 ㅊ 제과 ㅈ 제/out/production/운 체 과 제
❯ clear
 12:51:42 PM  -44  hyungwook  ~/Desktop/학 ㄱ 교강 ㅇ 영운 ㅇ 체제 ㅈ 제/운 ㅊ
제과 ㅈ 제2/운 ㅊ 제과 ㅈ 제/out/production/운 체 과 제
❯ java Main
CPU Burst Time: Job 1 < Job 2 < Job 3 < Job 4
============ FCFS ============
---------- Gantt chart ----------
Job 4 starting time: 0
Job 3 starting time: 410
Job 2 starting time: 717
Job 1 starting time: 729
-----------------------------
Job 4 waiting time: 0
Job 3 waiting time: 410
Job 2 waiting time: 717
Job 1 waiting time: 729
---> average waiting time: 464.0
============ SJF ============
---------- Gantt chart ----------
Job 1 starting time: 0
Job 2 starting time: 6
Job 3 starting time: 18
Job 4 starting time: 37
-----------------------------
Job 1 waiting time: 0
Job 2 waiting time: 6
Job 3 waiting time: 18
Job 4 waiting time: 37
---> average waiting time: 15.25
============ RR ============
timeSlice: 13
---------- Gantt chart ----------
Job 4 starting time: 0
Job 3 starting time: 14
Job 2 starting time: 26
Job 1 starting time: 39
Job 4 starting time: 52
Job 3 starting time: 65
Job 2 starting time: 78
Job 4 starting time: 91
-----------------------------
Job 4 waiting time: 65
Job 3 waiting time: 52
Job 2 waiting time: 65
Job 1 waiting time: 39
---> average waiting time: 55.25

The Scheduling Algorithm Evaluation end
 12:51:45 PM  -44  hyungwook  ~/Desktop/학 ㄱ 교강 ㅇ 영운 ㅇ 체제 ㅈ 제/운 ㅊ
제과 ㅈ 제2/운 ㅊ 제과 ㅈ 제/out/production/운 체 과 제
❯
```