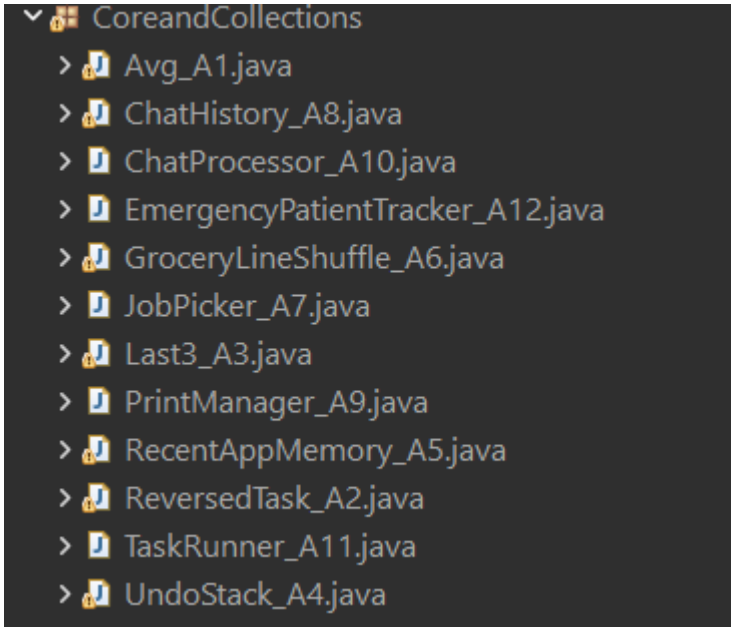
Core java and Collections



Avg_A1.java

```java
package Jagadeesh;
import java.util.Scanner;

public class Avg_A1 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] num = new int[5];
        int sum = 0;
        System.out.println("Enter the 5 Numbers");
        for(int i=0;i<5;i++) {
            num[i] = sc.nextInt();
            if(num[i]<10) {
                num[i]= num[i]*2;
            }
            sum = sum+num[i];
        }

        double average = sum/5.0;
        System.out.println("Average is"+ " "+average);

    }

}
```

Console

```
<terminated> Avg_A1 [Java Application] C:\Users\Jagadeesh Dowluri\Downloads\spring-to
Enter the 5 Numbers
25
30
41
65
48
Average is 41.8
```

ReversedTask_A2.java

```java
package Jagadeesh;

import java.util.*;

public class ReversedTask_A2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<String> tasks = new LinkedList<>();

        System.out.println("Enter 4 tasks:");
        for (int i = 0; i < 4; i++) {
            String task = sc.nextLine();
            if (task.endsWith("1")) {
                tasks.addFirst(task);
            } else {
                tasks.addLast(task);
            }
        }

        System.out.println("Reversed Task List:");
        for (int i = tasks.size() - 1; i >= 0; i--) {
            System.out.println(tasks.get(i));
        }
    }
}
```

Console

```
<terminated> ReversedTask_A2 [Java Application] C:\Users\Jagadeesh Dowluri\Downloads\
Enter 4 tasks:
fresh
run
wake
sleep
Reversed Task List:
sleep
wake
run
fresh
```

Last3_A3.java

```java
package Jagadeesh;
import java.util.*;

public class Last3_A3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayDeque<String> searchBox = new ArrayDeque<>();

        for (int i = 0; i < 4; i++) {
            System.out.print("Enter search term: ");
            String term = sc.nextLine();

            if (searchBox.size() == 3) {
                searchBox.removeFirst();
            }

            searchBox.addLast(term);
        }

        System.out.println("Last 3 searches:");
        for (String term : searchBox) {
            System.out.println(term);
        }
    }
}
```

Console

<terminated> Last3_A3 [Java Application] C:\Users\Jagadeesh Dowluri\Downloads\spring-t

```
Enter search term: jagadet
Enter search term: ramu
Enter search term: simu
Enter search term: sures
Last 3 searches:
ramu
simu
sures
```

UndoStack_A4.java

```java
package Jagadeesh;
import java.util.*;
public class UndoStack_A4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Stack<String> commandStack = new Stack<>();

        for (int i=1;i<=3;i++) {
            System.out.println("Enter Command"+ i + " : ");
            String command = sc.nextLine();
            commandStack.push(command);

    }
            String pop = commandStack.pop();
            System.out.println("Undo :" + pop);
            commandStack.push(pop);
            System.out.println("Redo :" + pop);
            System.out.println("Final Stack:");

            for(String command: commandStack) {
                System.out.println(command);
            }

        }
    }
```

Console

<terminated> UndoStack_A4 [Java Application] C:\Users\Jagadeesh Dowluri\Downloads\sp

```
Enter Command1 :
Run
Enter Command2 :
Slow
Enter Command3 :
Fast
Undo :Fast
Redo :Fast
Final Stack:
Run
Slow
Fast
```

RecentAppMemory_A5.java

```java
package Jagadeesh;
import java.util.*;

public class RecentAppMemory_A5{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<String> apps = new LinkedList<>();

        System.out.println("Enter 5 app names:");
        for (int i = 0; i < 5; i++) {
            String app = sc.nextLine();
            if (apps.contains(app)) {
                apps.remove(app);
            }
            apps.addFirst(app);
        }

        System.out.println("Final app memory list:");
        for (String app : apps) {
            System.out.println(app);
        }
    }
}
```

Console

```
<terminated> RecentAppMemory_A5 [Java Application] C:\Users\Jagadeesh Dowluri\Down
Enter 5 app names:
Whatsapp
facebook
tiktok
insta
twitter
Final app memory list:
twitter
insta
tiktok
facebook
Whatsapp
```

GroceryLineShuffle_A6.java

```java
package Jagadeesh;

import java.util.*;

public class GroceryLineShuffle_A6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayDeque<String> queue = new ArrayDeque<>();

        System.out.println("Enter 5 customer names:");
        for (int i = 0; i < 5; i++) {
            String name = sc.nextLine();
            if (name.length() % 2 == 0) {
                queue.addFirst(name);
            } else {
                queue.addLast(name);
            }
        }

        System.out.println("Final queue order:");
        for (String name : queue) {
            System.out.println(name);
        }
    }
}
```

Console

```
<terminated> GroceryLineShuffle_A6 [Java Application] C:\Users\Jagadeesh Dowluri\Down
Enter 5 customer names:
Ramu
somu
srinu
prabhu
rana
Final queue order:
rana
prabhu
somu
Ramu
srinu
```

JobPicker_A7.java

```java
public class JobPicker_A7 {
    private PriorityQueue<Job> jobQueue;

    public JobPicker_A7() {
        jobQueue = new PriorityQueue<>((job1, job2) -> {
            if (job1.getUrgency() != job2.getUrgency()) {
                return Integer.compare(job1.getUrgency(), job2.getUrgency());
            }
            return Integer.compare(job1.getName().length(), job2.getName().length());
        });
    }

    public void addJob(String name, int urgency) {
        if (urgency < 1 || urgency > 5) {
            throw new IllegalArgumentException("Urgency must be between 1 and 5.");
        }
        jobQueue.offer(new Job(name, urgency));
    }

    public Job getNextJob() {
        return jobQueue.poll();
    }

    public static void main(String[] args) {
        JobPicker_A7 scheduler = new JobPicker_A7();
        scheduler.addJob("Fix bug", 2);
        scheduler.addJob("Write tests", 3);
        scheduler.addJob("Implement feature", 1);
        scheduler.addJob("Code review", 2);
        scheduler.addJob("Deploy", 5);
        scheduler.addJob("Update documentation", 2);

        Job nextJob;
        while ((nextJob = scheduler.getNextJob()) != null) {
            System.out.println("Next job: " + nextJob.getName() + " (Urgency: " + nextJob.getUrgency() + ")");
        }
    }
}
```

Console

```
<terminated> JobPicker_A7 [Java Application] C:\Users\Jagadeesh Dowluri\Downloads\spr
Next job: Implement feature (Urgency: 1)
Next job: Fix bug (Urgency: 2)
Next job: Code review (Urgency: 2)
Next job: Update documentation (Urgency: 2)
Next job: Write tests (Urgency: 3)
Next job: Deploy (Urgency: 5)
```

*ChatHistory_A8.java

```java
package Jagadeesh;

import java.util.*;

public class ChatHistory_A8 {
    public static void main(String[] args) {
        ArrayDeque<String> chatBox = new ArrayDeque<>();

        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.print("Enter message (type 'exit' to quit): ");
            String msg = sc.nextLine();

            if (msg.equalsIgnoreCase("exit")) break;

            if (chatBox.size() == 4) {
                chatBox.removeFirst();
            }

            chatBox.addLast(msg);

            System.out.println("Current Chat History:");
            for (String m : chatBox) {
                System.out.println(m);
            }
        }
    }
}
```

Console

```
<terminated> ChatHistory_A8 [Java Application] C:\Users\Jagadeesh Dowluri\Dow
Enter message (type 'exit' to quit): hello ram
Current Chat History:
hello ram
Enter message (type 'exit' to quit): how are you ?
Current Chat History:
hello ram
how are you ?
Enter message (type 'exit' to quit): are you there ?
Current Chat History:
hello ram
how are you ?
are you there ?
Enter message (type 'exit' to quit): exit
```

PrintManager_A9.java

```java
package Jagadeesh;

import java.util.concurrent.ArrayBlockingQueue;

public class PrintManager_A9 {
    public static void main(String[] args) {
        ArrayBlockingQueue<String> printQueue = new ArrayBlockingQueue<>(3);


        addJob(printQueue, "Job1");
        addJob(printQueue, "Job2");
        addJob(printQueue, "Job3");
        addJob(printQueue, "Job4");

        while (!printQueue.isEmpty()) {
            System.out.println("Printing: " + printQueue.poll());
        }
    }

    public static void addJob(ArrayBlockingQueue<String> queue, String jobName) {
        if (queue.offer(jobName)) {
            System.out.println("Added to queue: " + jobName);
        } else {
            System.out.println("Queue full! Skipped: " + jobName);
        }
    }
}

```

Console

```
<terminated> PrintManager_A9 [Java Application] C:\Users\Jagadees
Added to queue: Job1
Added to queue: Job2
Added to queue: Job3
Queue full! Skipped: Job4
Printing: Job1
Printing: Job2
Printing: Job3
```

ChatProcessor_A10.java

```java
{
    public static void main(String[] args) {
        LinkedBlockingQueue<String> chatQueue = new LinkedBlockingQueue<>();

        Thread producer = new Thread(() -> {
            int count = 1;
            while (true) {
                try {
                    if (chatQueue.size() < 5) {
                        String msg = "Message " + count++;
                        chatQueue.put(msg);
                        System.out.println("Produced: " + msg);
                    } else {
                        System.out.println("Queue limit reached. Producer is waiting.
                        Thread.sleep(1000);
                    }
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        Thread consumer = new Thread(() -> {
            while (true) {
                try {
                    String msg = chatQueue.take();
                    System.out.println("Consumed: " + msg);
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        producer.start();
        consumer.start();
    }
}

```

Console

```
ChatProcessor_A10 [Java Application] C:\Users\Jagadeesh Dowluri\Dow
Queue limit reached. Producer is waiting...
Consumed: Message 25
Consumed: Message 26
Produced: Message 30
Produced: Message 31
Consumed: Message 27
Produced: Message 32
Queue limit reached. Producer is waiting...
Consumed: Message 28
Consumed: Message 29
Produced: Message 33
Produced: Message 34
Consumed: Message 30
Produced: Message 35
Queue limit reached. Producer is waiting...
Consumed: Message 31
```

TaskRunner_A11.java

```java
import java.util.concurrent.LinkedBlockingQueue;

class Task {
    int id;
    String description;

    Task(int id, String description) {
        this.id = id;
        this.description = description;
    }

    @Override
    public String toString() {
        return "Task ID: " + id + ", Desc: " + description;
    }
}

public class TaskRunner_A11 {
    public static void main(String[] args) {
        LinkedBlockingQueue<Task> stage1 = new LinkedBlockingQueue<>();
        LinkedBlockingQueue<Task> stage2 = new LinkedBlockingQueue<>();


        for (int i = 1; i <= 6; i++) {
            stage1.add(new Task(i, "Task number " + i));
        }


        while (!stage1.isEmpty()) {
            Task task = stage1.poll();
            System.out.println("Stage 1 processed: " + task);

            if (task.id % 2 == 0) {
                stage2.add(task);
            }
        }


        System.out.println("\n--- Stage 2 Tasks ---");
        while (!stage2.isEmpty()) {
            Task task = stage2.poll();
            System.out.println("Stage 2 executed: " + task);
        }
    }
}
```

Console

```
<terminated> TaskRunner_A11 [Java Application] C:\Users\Jagadeesh Dowluri\Dov
Stage 1 processed: Task ID: 1, Desc: Task number 1
Stage 1 processed: Task ID: 2, Desc: Task number 2
Stage 1 processed: Task ID: 3, Desc: Task number 3
Stage 1 processed: Task ID: 4, Desc: Task number 4
Stage 1 processed: Task ID: 5, Desc: Task number 5
Stage 1 processed: Task ID: 6, Desc: Task number 6

--- Stage 2 Tasks ---
Stage 2 executed: Task ID: 2, Desc: Task number 2
Stage 2 executed: Task ID: 4, Desc: Task number 4
Stage 2 executed: Task ID: 6, Desc: Task number 6
```

EmergencyPatientTracker_A12.java

```java
package Jagadeesh;

import java.util.*;

class Patient {
    String name;
    int severity;
    long timestamp;

    Patient(String name, int severity) {
        this.name = name;
        this.severity = severity;
        this.timestamp = System.currentTimeMillis();
    }

    @Override
    public String toString() {
        return name + " (Severity: " + severity + ")";
    }
}

public class EmergencyPatientTracker_A12 {
    public static void main(String[] args) throws InterruptedException {
        Comparator<Patient> comparator = (p1, p2) -> {
            if (p1.severity != p2.severity)
                return Integer.compare(p1.severity, p2.severity);
            else
                return Long.compare(p1.timestamp, p2.timestamp);
        };

        PriorityQueue<Patient> queue = new PriorityQueue<>(comparator);


        addPatient(queue, new Patient("Alice", 3));
        Thread.sleep(100);
        addPatient(queue, new Patient("Bob", 1));
        Thread.sleep(100);
        addPatient(queue, new Patient("Charlie", 2));
        Thread.sleep(100);
        addPatient(queue, new Patient("David", 1));
        Thread.sleep(100);
        addPatient(queue, new Patient("Eve", 2));
        Thread.sleep(100);
        addPatient(queue, new Patient("Frank", 4));
```

Console

```
<terminated> EmergencyPatientTracker_A12 [Java Application] C:\Users\Jagadeesh
Added: Alice (Severity: 3)
Added: Bob (Severity: 1)
Added: Charlie (Severity: 2)
Added: David (Severity: 1)
Added: Eve (Severity: 2)
Queue full. Skipped: Frank (Severity: 4)

--- Treating Patients ---
Treated: Bob (Severity: 1)
Treated: David (Severity: 1)
Treated: Charlie (Severity: 2)
Treated: Eve (Severity: 2)
Treated: Alice (Severity: 3)
```