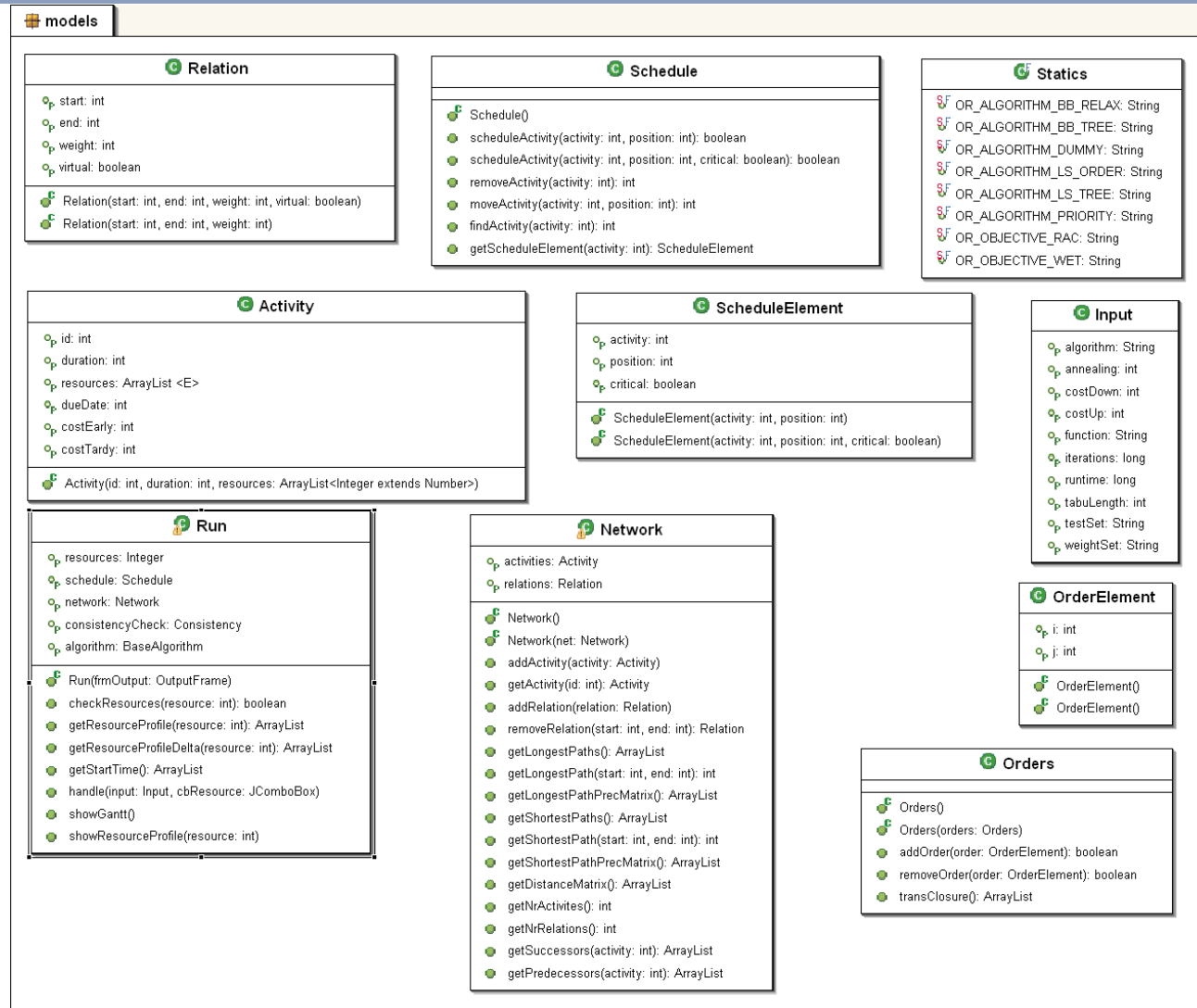


# Zwischenpräsentation

## Gruppe: Integrationsumgebung

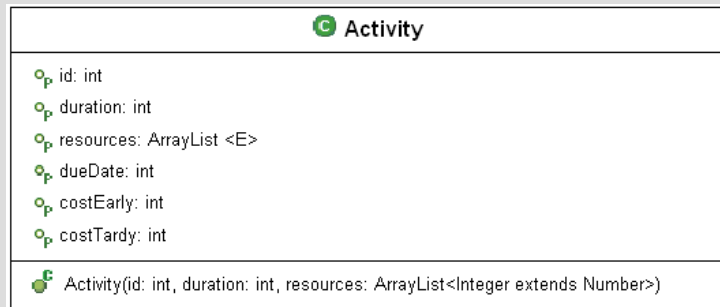
Jaro Leitner  
 Markus Maier  
 Bernhard Schwarz

# Übersicht Datenmodell



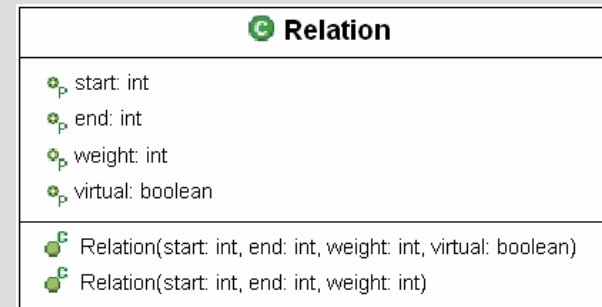
# Repräsentation der Baumstruktur

## Activity



- Klasse zur Speicherung der Aktivitäten (Knoten im Graphen)
- Enthält die Elemente
  - Nummer der Aktivität
  - Dauer der Aktivität
  - Benötigte Ressourcen
  - Due-Date der Aktivität
  - Kosten der Aktivität bei WET

## Relation



- Klasse zur Speicherung der Beziehungen (Kanten im Graphen)
- Enthält die Elemente
  - Nummer der Startaktivität der Kante
  - Nummer der Endaktivität der Kante
  - Gewicht der Kante
  - Attribut virtual (z.B. für während des Algorithmus hinzugefügte Kanten)

# Darstellung der geplanten Aktivitäten

## ScheduleElement

### ☐ ScheduleElement

• activity: int  
• position: int  
• critical: boolean

☐ ScheduleElement(activity: int, position: int)  
☐ ScheduleElement(activity: int, position: int, critical: boolean)

- Klasse zur Speicherung geplanter Aktivitäten im Zeitplan
- Enthält die Elemente
  - Nummer der Aktivität
  - Startzeitpunkt
  - Attribut kritisch (für Elemente, die nicht zu einem späteren Zeitpunkt gestartet werden dürfen)

## Schedule

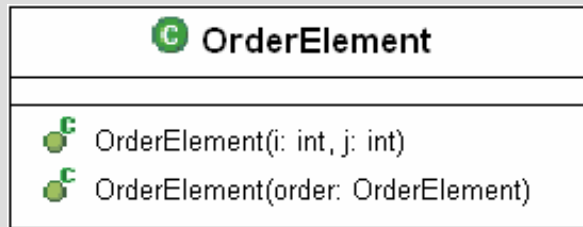
### ☐ Schedule

☐ Schedule()  
• scheduleActivity(activity: int, position: int): boolean  
• scheduleActivity(activity: int, position: int, critical: boolean): boolean  
• removeActivity(activity: int): int  
• moveActivity(activity: int, position: int): int  
• findActivity(activity: int): int  
• getScheduleElement(activity: int): ScheduleElement

- Liste zur Speicherung der einzelnen Schedule-Elemente
- Enthält die Methoden
  - scheduleActivity: Einfügen einer weiteren Aktivität in den Zeitplan
  - removeActivity: Entfernen einer Aktivität aus dem Zeitplan
  - moveActivity: Verschieben einer Aktivität im Zeitplan
  - findActivity: Ermitteln des Startzeitpunktes einer Aktivität
  - getScheduleElement: Gibt das Schedule-Element zur Aktivität aus

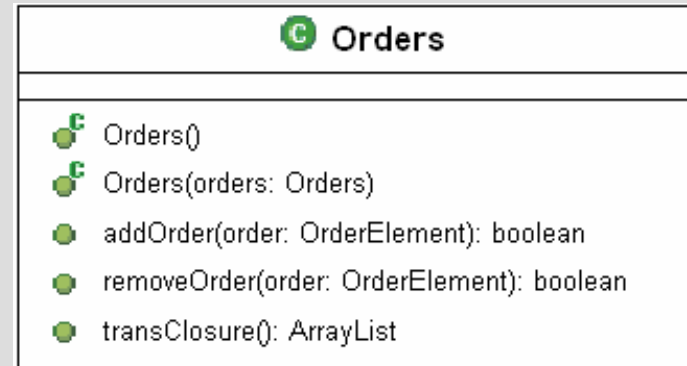
# Repräsentation beliebiger Ordnungen

## OrderElement













- Klasse zur Speicherung von Ordnungselementen
- Enthält die Nummern zweier Ordnungselemente

## Orders



- Liste zur Speicherung der einzelnen Ordnungselemente
- Enthält die Methoden
  - addOrder: Hinzufügen eines Ordnungselementes
  - removeOrder: Entfernen eines Ordnungselementes
  - transClosure: Ermittlung der transitiven Hülle der Ordnung

## Input



-  algorithm: String
-  annealing: int
-  costDown: int
-  costUp: int
-  function: String
-  iterations: long
-  runtime: long
-  tabuLength: int
-  testSet: String
-  weightSet: String


## Input

- Klasse zur Verwaltung aller Eingaben
- Enthält die Elemente
  - Gewählter Algorithmus
  - Abkühlungsparameter für das Simulated Annealing
  - Kosten der Ressourcen-Änderungen (Sprünge im Ressourcen-Profil)
  - Gewählte Zielfunktion
  - Anzahl vorgegebener Iterationen
  - Vorgegebene Dauer der Algorithmusausführung
  - Länge der Tabuliste
  - Dateiname des Test-Sets
  - Dateiname der WET-Gewichte

# Verwaltung eines Netzwerkes

## Network

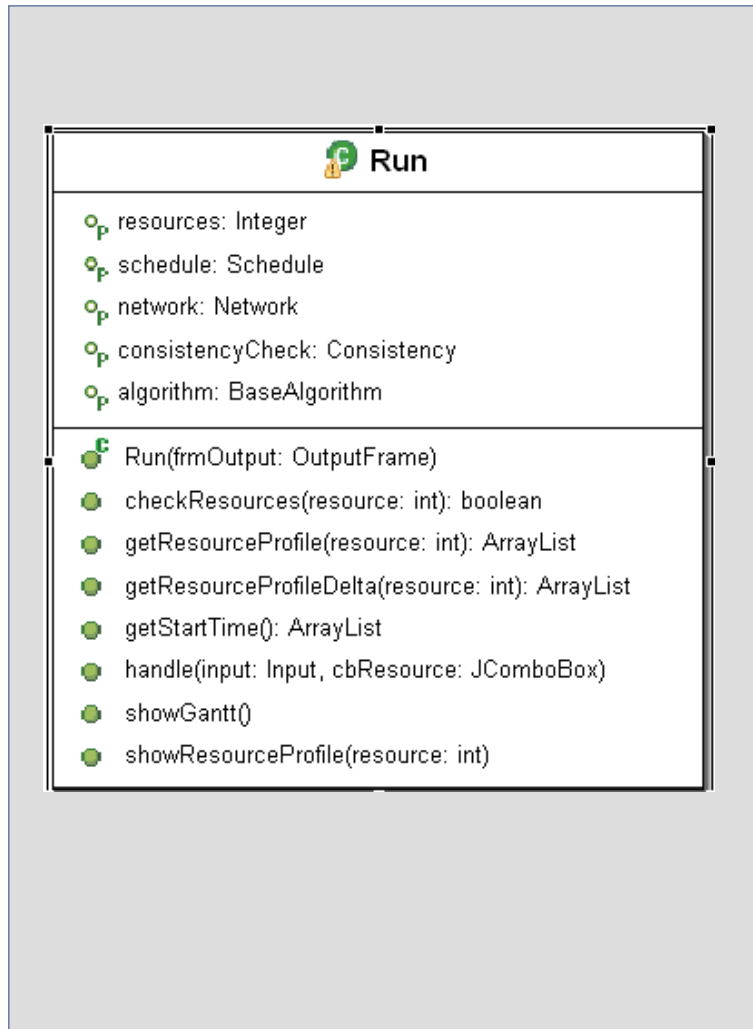
 activities: Activity  
 relations: Relation

 Network()  
 Network(net: Network)  
 addActivity(activity: Activity)  
 getActivity(id: int): Activity  
 addRelation(relation: Relation)  
 removeRelation(start: int, end: int): Relation  
 getLongestPaths(): ArrayList  
 getLongestPath(start: int, end: int): int  
 getLongestPathPrecMatrix(): ArrayList  
 getShortestPaths(): ArrayList  
 getShortestPath(start: int, end: int): int  
 getShortestPathPrecMatrix(): ArrayList  
 getDistanceMatrix(): ArrayList  
 getNrActivites(): int  
 getNrRelations(): int  
 getSuccessors(activity: int): ArrayList  
 getPredecessors(activity: int): ArrayList

## Network

- Klasse zur Verwaltung von Netzwerken
- Enthält die Methoden
  - addActivity: Hinzufügen einer Aktivität (Knoten) im Graphen
  - getActivity: Rückgabe einer Aktivität
  - addRelation: Hinzufügen einer Relation (Kante) im Graphen
  - removeRelation: Entfernen einer Relation im Graphen
  - getLongestPaths: Ermittlung der Längste-Wege-Matrix
  - getLongestPath: Ermittlung des längsten Weges zwischen zwei Knoten bzw. der gesamten Längste-Wege-Matrix
  - getLongestPathPrecMatrix: Matrix der Vorgänger auf längstem Weg
  - getDistanceMatrix: Bestimmung der Distanzmatrix
  - getNrActivites: gibt die Anzahl der Aktivitäten zurück
  - getNrRelations: gibt die Anzahl der Relationen zurück
  - getSuccessors: Liste der Nachfolger
  - getPredecessors: Liste der Vorgänger

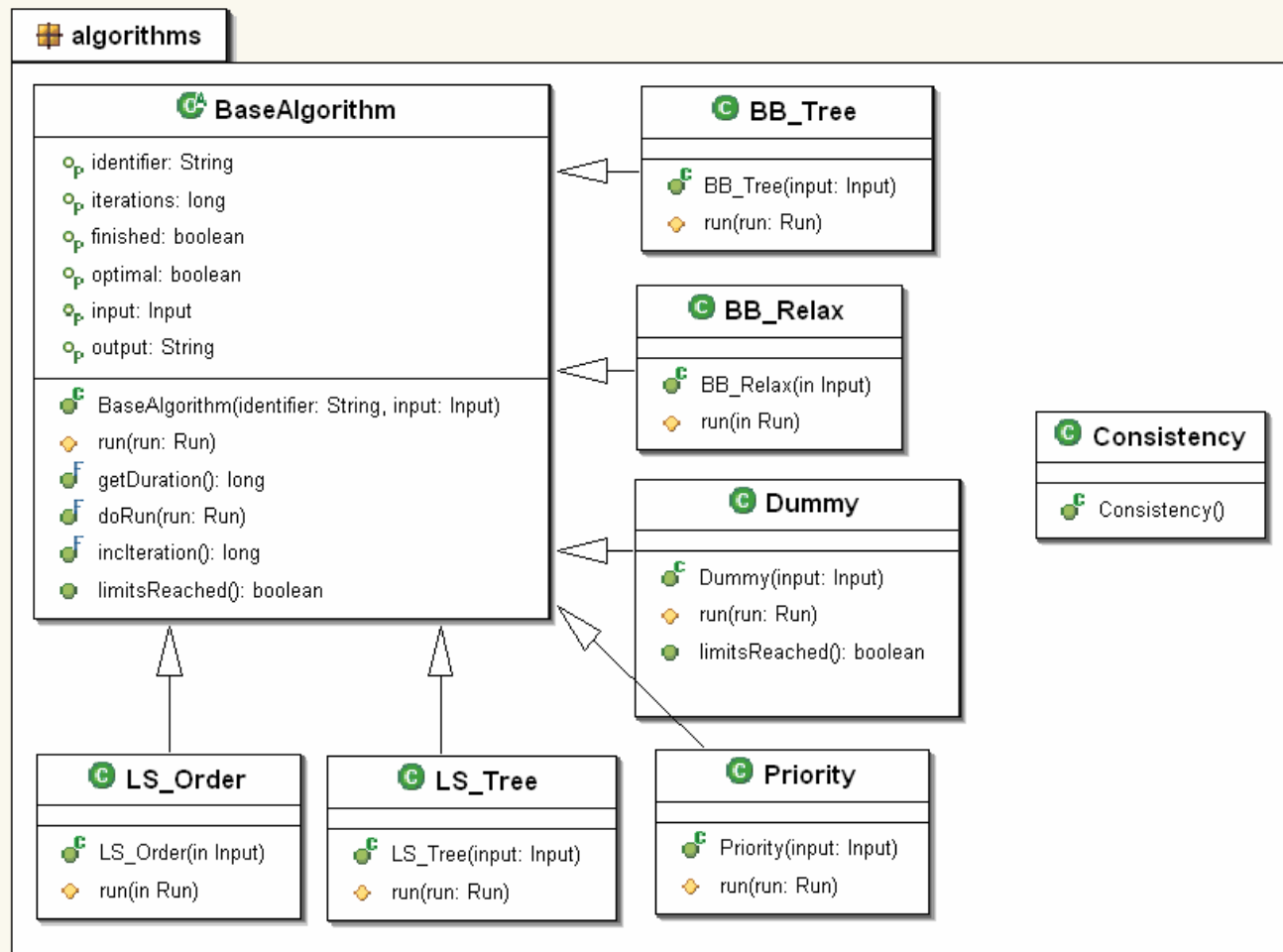
# Verwaltung eines Simulationslaufes



## Run

- Klasse zur Verwaltung aller allgemeinen Methoden für einen Simulationslauf
- Enthält die Methoden
  - checkRessources: Überprüfung der Ressourcen-Beschränkung
  - getResourceProfile: Ermittlung des Ressourcen-Profiles
  - getResourceProfileDelta: Ermittlung der Sprünge im Ressourcen-Profil
  - getStartTime: Liste der Startzeitpunkte der Aktivitäten
  - showGantt: Ausgabe des Gantt-Charts
  - showResourceProfile: Ausgabe des Ressourcen-Profiles

# Struktur der Algorithmus-Klasse und den Basis-Methoden



## Darstellung einer beispielhaften Algorithmusstruktur

```
/**
 * Standardkonstruktor
 */
public Dummy(Input input) {
    super(OR_ALGORITHM_DUMMY, input);
}

/**
 * falls weitere Limits geprüft werden müssen
 */
public boolean limitsReached(){
    boolean kriterium = false;

    return(kriterium || super.limitsReached());
}
```

```
/**
 * startet den eigentlichen Algorithmus
 */
protected void run(Run run) {

    // ein paar Aktivitäten planen
    run.getSchedule().scheduleActivity(1, 4);
    run.getSchedule().scheduleActivity(2, 3);
    run.getSchedule().scheduleActivity(6, 9);
    run.getSchedule().scheduleActivity(9, 12);

    // ein paar mal durchlaufen lassen
    for (int i = 0; i < 1000; i++) {
        if (!this.limitsReached()) {
            this.incIteration();
        }
    }

    // ein bisschen warten
    try {
        Thread.sleep(1000);
    } catch (Exception e) {
        System.out.println(e);
        System.exit(1);
    }

    // Algorithmus hat optimales Ergebnis gefunden
    this.setOptimal(true);

    // Algorithmus wurde nicht durch Limits abgebrochen
    if (!this.limitsReached()) {
        this.setFinished(true);
    }
}
```

## Zu Implementierende Algorithmen

### Algorithmen

1. Branch & Bound Relaxation-Based
2. Branch & Bound Tree-Based
3. Local Search Order Based
4. Local Search Tree Based
5. Untere Schranken & Konsistenztests
6. Prioritätsregelverfahren

### Zielfunktionen

1. Weighted Earliest Tardiness
2. Resource Adjustment Costs

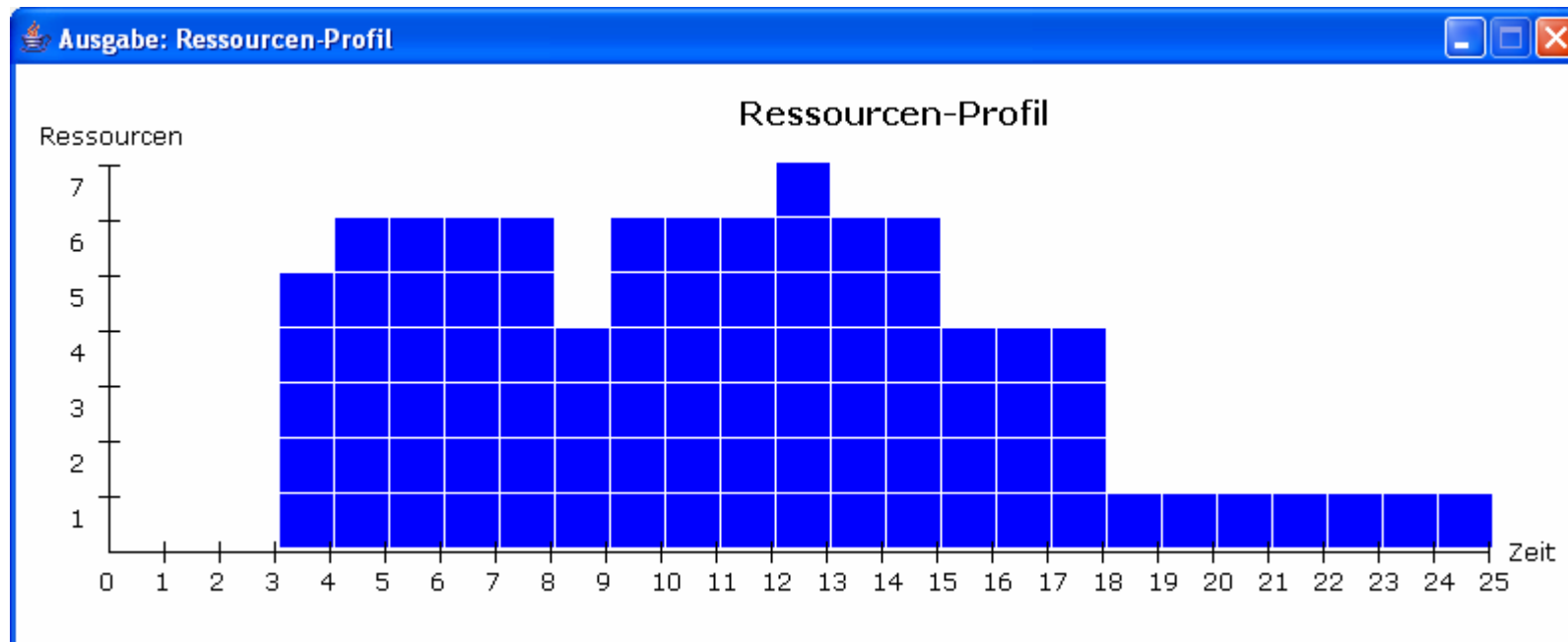
# Screenshot der Oberfläche

The screenshot shows the 'OR-Softwarepraktikum Simulator' window. It contains the following configuration options:

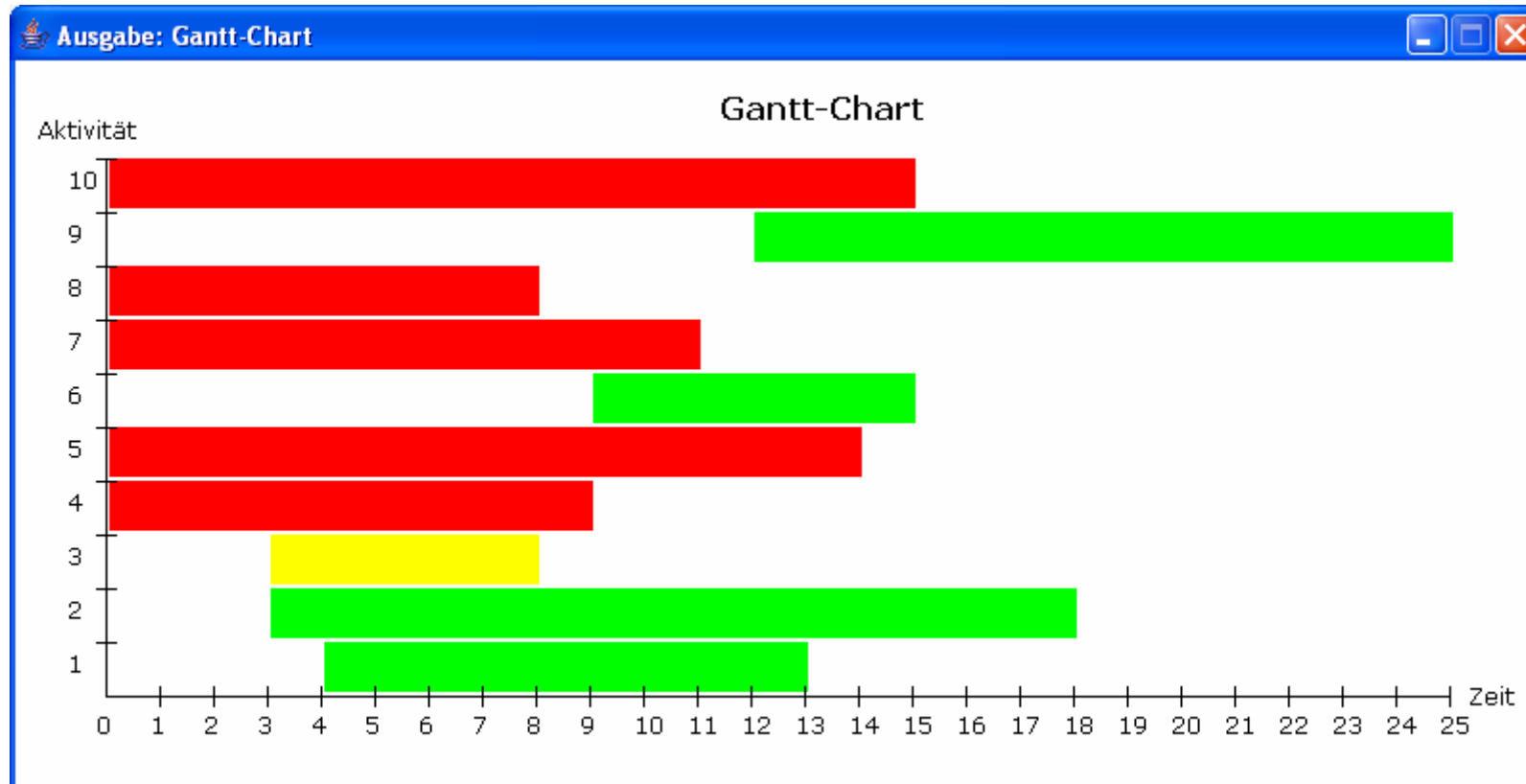
- Algorithmus:** Dummy Algorithmus
- Zielfunktion:** Ressource Adjustment Costs
- Testset:** E:\Uni\OR Software Praktikum\Scheduler\data (wählen)
- Gewichte:** E:\Uni\OR Software Praktikum\Scheduler\data (wählen)
- Iterationen:** 0
- Laufzeit [s]:** 0
- Implementationskosten:** auf 3, ab 2
- Länge Tabu-Liste:** 3
- Abkühlungsparameter:** 2

Buttons at the bottom include 'Starten', a dropdown menu, 'Ressourcen', and 'Gantt'.

# Screenshot der Ausgabe des Ressourcen-Profiles



# Screenshot der Ausgabe des Gantt-Charts



- Nicht eingeplante Aktivität
- Eingeplante zeitkritische Aktivität
- Eingeplante Aktivität

## Offene Punkte und weiteres Vorgehen

### Offene Punkte

1. Wurden alle Verfahren in der Auswahl korrekt erfasst und sinnvoll geclustert?
2. Fehlen wesentliche Elemente in der Datenstruktur oder wichtige Methoden?
3. Werden die Konsistenztests nur zu Beginn der anderen Verfahren aufgerufen oder auch zu späteren Zeitpunkten (z.B. im B&B)?

### Ausblick

Bis zum kommenden Dienstag, den 17.01.2006 werden alle erörterten Änderungswünsche eingearbeitet und das Gerüst wird allen Gruppen zur Verfügung gestellt.