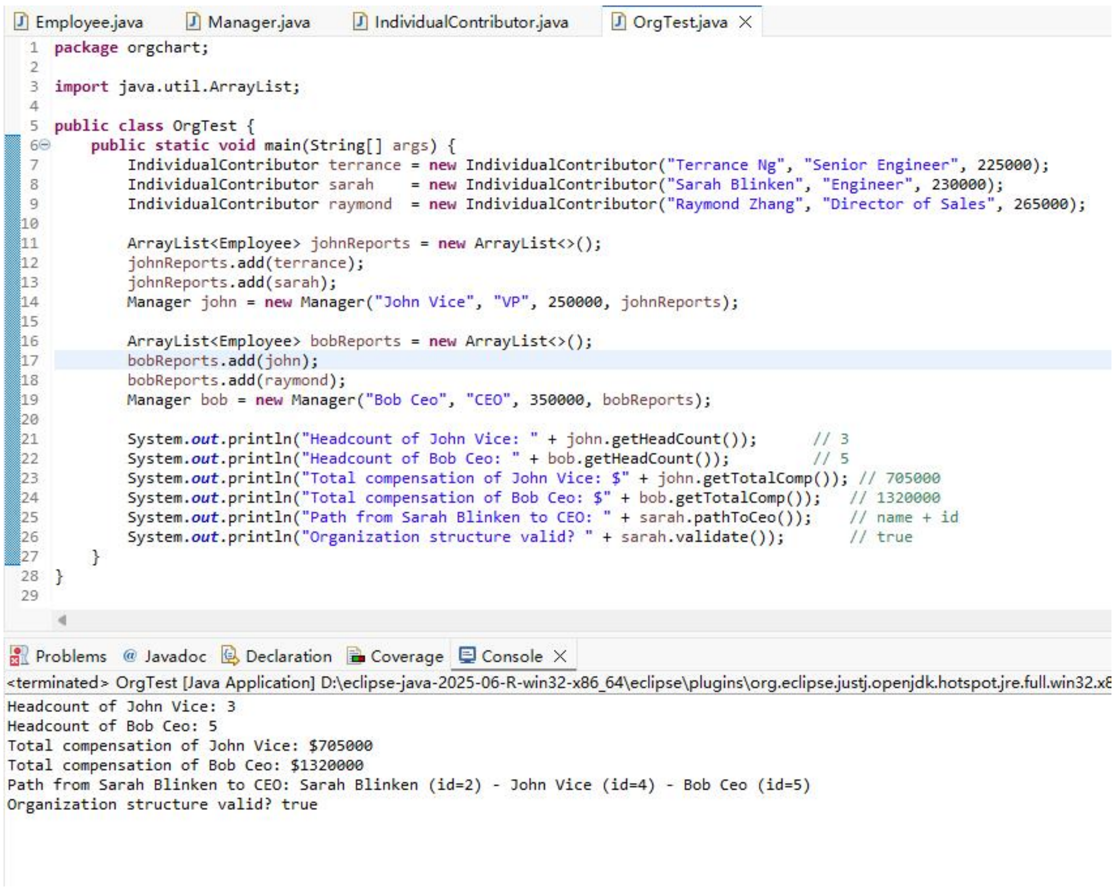
Part1

The org chart implementation demonstrates the use of inheritance, abstraction, and recursive computation in building a hierarchical employee management model. The design centers on an abstract `Employee` superclass that encapsulates shared attributes such as ID, name, title, and salary, while specialized subclasses `Manager` and `IndividualContributor` extend functionality to represent leadership and non-leadership roles. Through recursive traversal, the program dynamically calculates organizational headcount and total compensation within any managerial subtree. A validation method ensures structural integrity by verifying bidirectional manager-report relationships and detecting potential cycles. The resulting hierarchy accurately models reporting lines from individual contributors to the CEO and produces consistent analytical outputs for hierarchy size, compensation aggregation, and upward navigation paths.

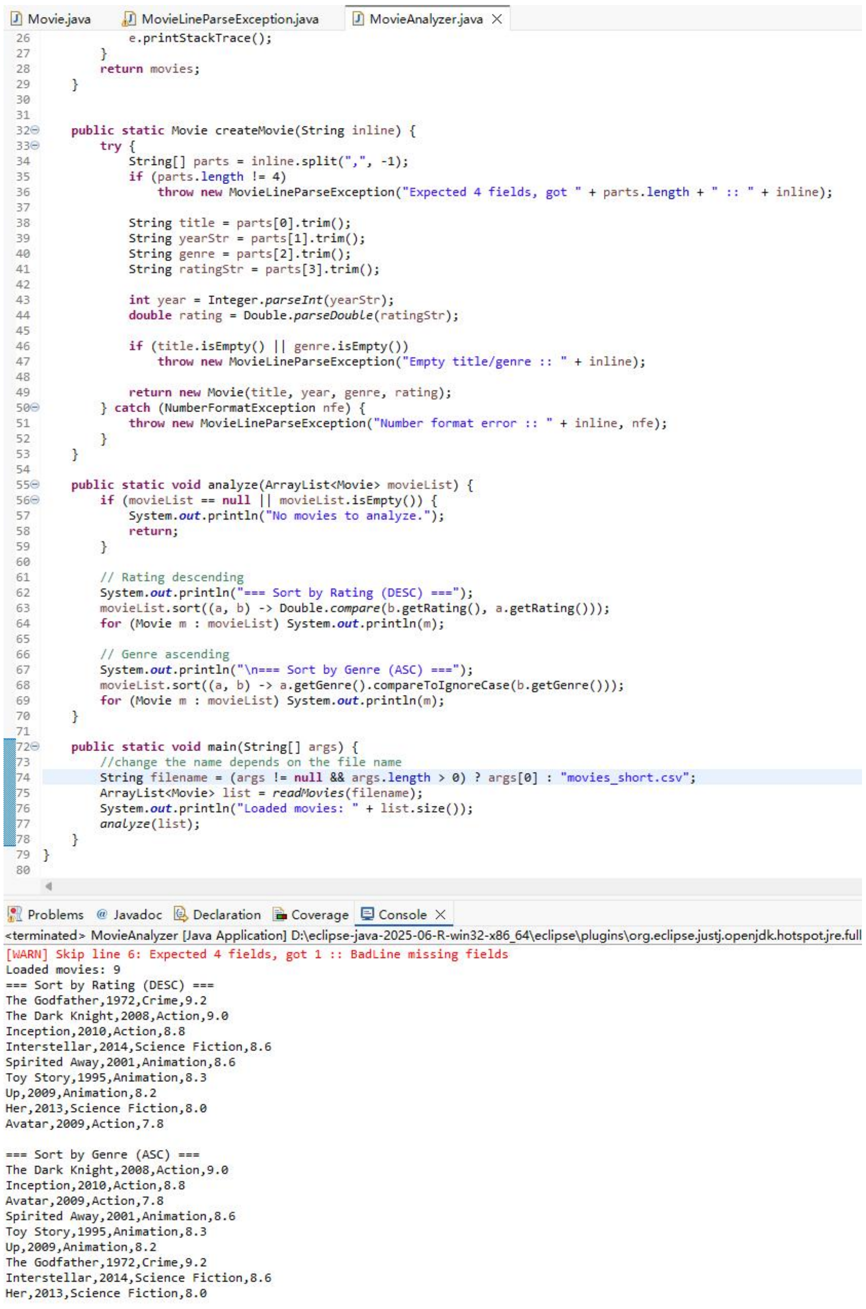


```java
package orgchart;

import java.util.ArrayList;

public class OrgTest {
    public static void main(String[] args) {
        IndividualContributor terrance = new IndividualContributor("Terrance Ng", "Senior Engineer", 225000);
        IndividualContributor sarah    = new IndividualContributor("Sarah Blinken", "Engineer", 230000);
        IndividualContributor raymond  = new IndividualContributor("Raymond Zhang", "Director of Sales", 265000);

        ArrayList<Employee> johnReports = new ArrayList<>();
        johnReports.add(terrance);
        johnReports.add(sarah);
        Manager john = new Manager("John Vice", "VP", 250000, johnReports);

        ArrayList<Employee> bobReports = new ArrayList<>();
        bobReports.add(john);
        bobReports.add(raymond);
        Manager bob = new Manager("Bob Ceo", "CEO", 350000, bobReports);

        System.out.println("Headcount of John Vice: " + john.getHeadCount());      // 3
        System.out.println("Headcount of Bob Ceo: " + bob.getHeadCount());         // 5
        System.out.println("Total compensation of John Vice: $" + john.getTotalComp()); // 705000
        System.out.println("Total compensation of Bob Ceo: $" + bob.getTotalComp());   // 1320000
        System.out.println("Path from Sarah Blinken to CEO: " + sarah.pathToCeo());    // name + id
        System.out.println("Organization structure valid? " + sarah.validate());      // true
    }
}
```

```
<terminated> OrgTest [Java Application] D:\eclipse-java-2025-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
Headcount of John Vice: 3
Headcount of Bob Ceo: 5
Total compensation of John Vice: $705000
Total compensation of Bob Ceo: $1320000
Path from Sarah Blinken to CEO: Sarah Blinken (id=2) - John Vice (id=4) - Bob Ceo (id=5)
Organization structure valid? true
```

Part2

The movie analyzer showcases robust file parsing and functional data processing using exception handling and lambda-based sorting. It reads a CSV dataset into a structured list of Movie objects, gracefully handling malformed records through a custom unchecked exception while maintaining processing continuity. The analyzer applies functional programming principles to sort and display the dataset by different criteria—descending ratings and ascending genres—reflecting both analytical precision and code efficiency. This implementation highlights practical data-cleaning resilience, modular decomposition between parsing and analysis, and the expressive power of Java' s lambda syntax for flexible dataset manipulation.



```java
            e.printStackTrace();
        }
        return movies;
    }


    public static Movie createMovie(String inline) {
        try {
            String[] parts = inline.split(",", -1);
            if (parts.length != 4)
                throw new MovieLineParseException("Expected 4 fields, got " + parts.length + " :: " + inline);

            String title = parts[0].trim();
            String yearStr = parts[1].trim();
            String genre = parts[2].trim();
            String ratingStr = parts[3].trim();

            int year = Integer.parseInt(yearStr);
            double rating = Double.parseDouble(ratingStr);

            if (title.isEmpty() || genre.isEmpty())
                throw new MovieLineParseException("Empty title/genre :: " + inline);

            return new Movie(title, year, genre, rating);
        } catch (NumberFormatException nfe) {
            throw new MovieLineParseException("Number format error :: " + inline, nfe);
        }
    }

    public static void analyze(ArrayList<Movie> movieList) {
        if (movieList == null || movieList.isEmpty()) {
            System.out.println("No movies to analyze.");
            return;
        }

        // Rating descending
        System.out.println("=== Sort by Rating (DESC) ===");
        movieList.sort((a, b) -> Double.compare(b.getRating(), a.getRating()));
        for (Movie m : movieList) System.out.println(m);

        // Genre ascending
        System.out.println("\n=== Sort by Genre (ASC) ===");
        movieList.sort((a, b) -> a.getGenre().compareToIgnoreCase(b.getGenre()));
        for (Movie m : movieList) System.out.println(m);
    }

    public static void main(String[] args) {
        //change the name depends on the file name
        String filename = (args != null && args.length > 0) ? args[0] : "movies_short.csv";
        ArrayList<Movie> list = readMovies(filename);
        System.out.println("Loaded movies: " + list.size());
        analyze(list);
    }
}
```

```
<terminated> MovieAnalyzer [Java Application] D:\eclipse-java-2025-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
[WARN] Skip line 6: Expected 4 fields, got 1 :: BadLine missing fields
Loaded movies: 9
=== Sort by Rating (DESC) ===
The Godfather,1972,Crime,9.2
The Dark Knight,2008,Action,9.0
Inception,2010,Action,8.8
Interstellar,2014,Science Fiction,8.6
Spirited Away,2001,Animation,8.6
Toy Story,1995,Animation,8.3
Up,2009,Animation,8.2
Her,2013,Science Fiction,8.0
Avatar,2009,Action,7.8

=== Sort by Genre (ASC) ===
The Dark Knight,2008,Action,9.0
Inception,2010,Action,8.8
Avatar,2009,Action,7.8
Spirited Away,2001,Animation,8.6
Toy Story,1995,Animation,8.3
Up,2009,Animation,8.2
The Godfather,1972,Crime,9.2
Interstellar,2014,Science Fiction,8.6
Her,2013,Science Fiction,8.0
```