Task Management System

Design and Implementation

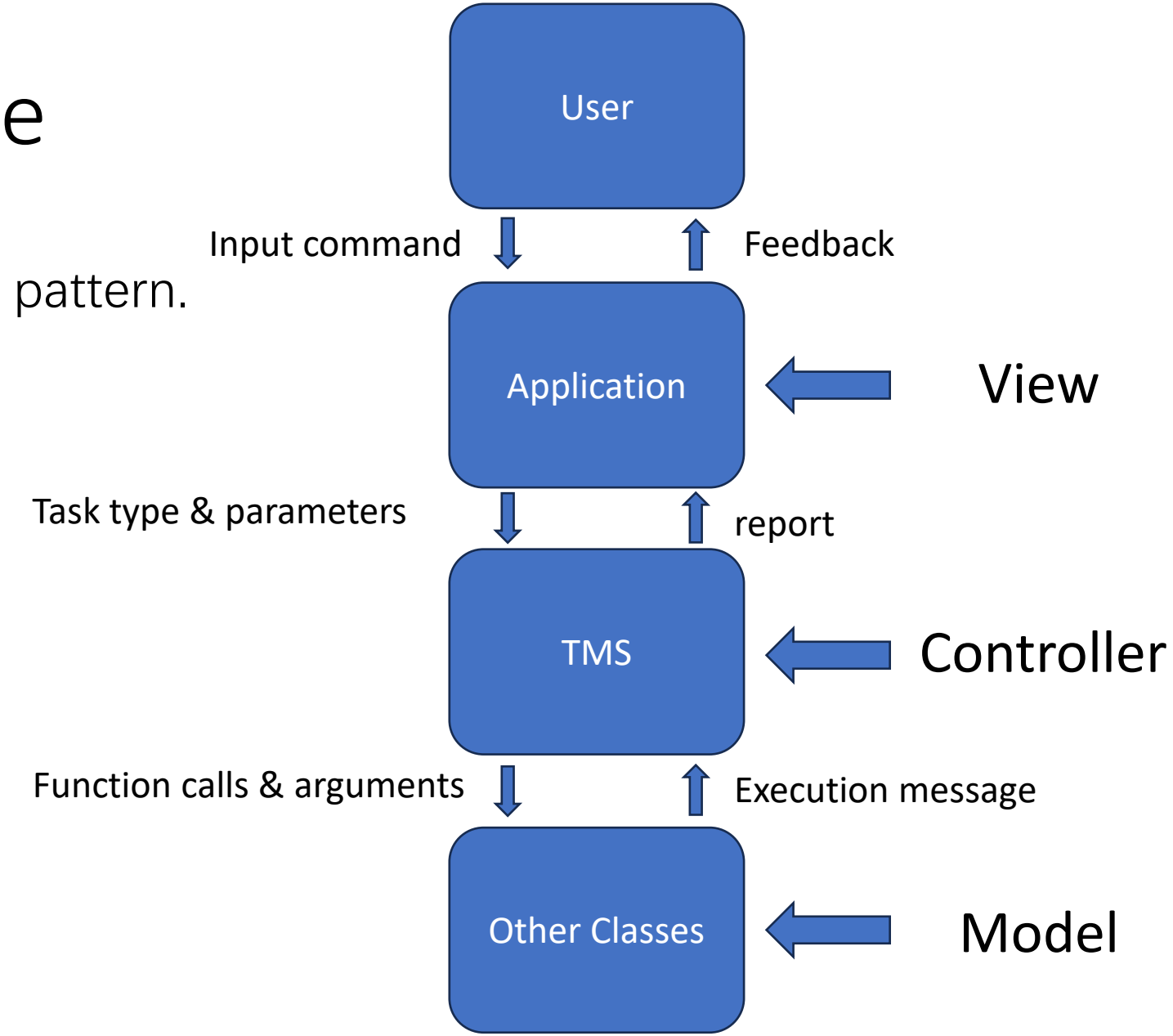
Presenter: LIN Zhanzhi, Eric

Introduction

- a user-friendly interface
- robust functionality
- Purpose:
  - enhance task management
  - improve productivity and efficiency.

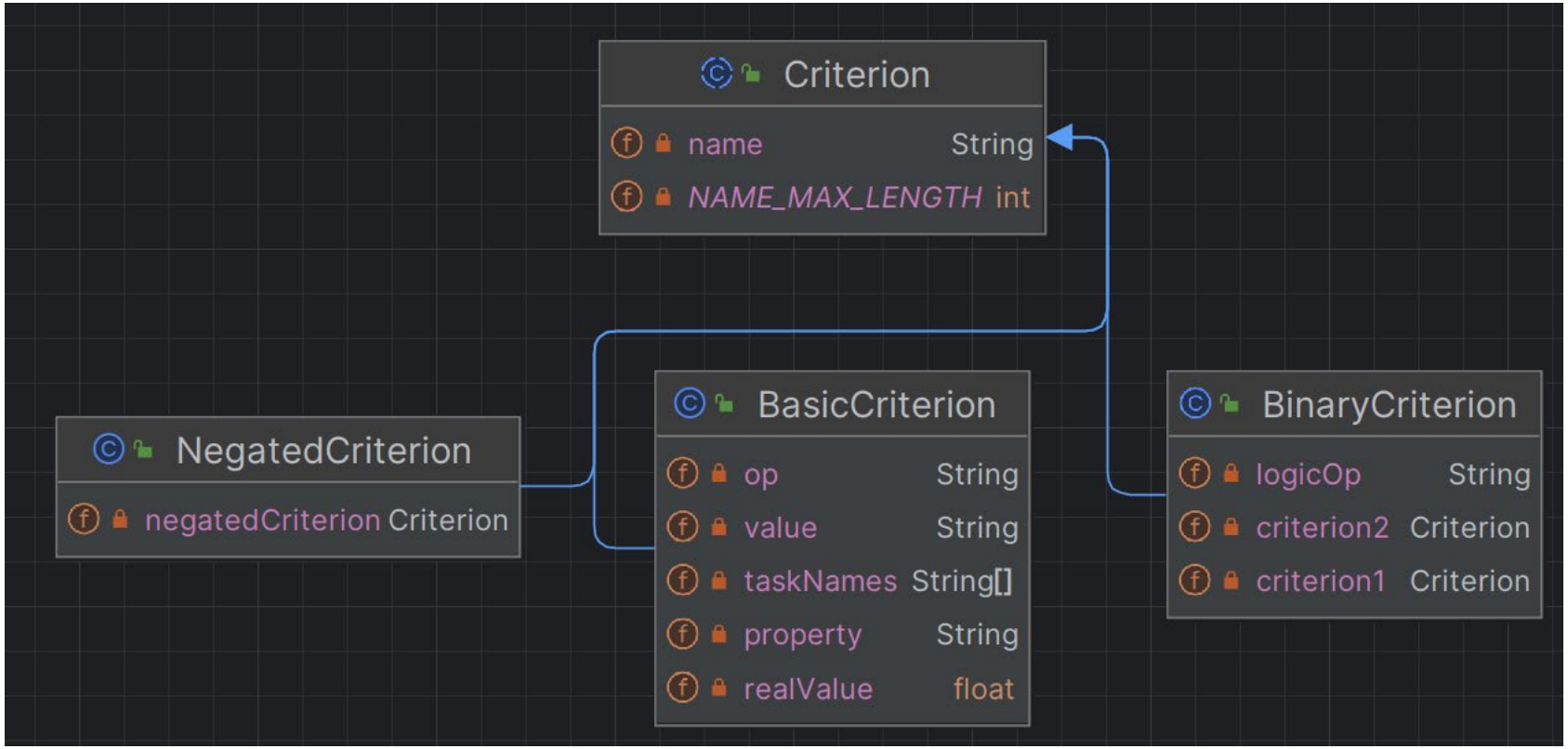
Overall Architecture

- Model-View-Controller (MVC) pattern.
- View: the Application class.
- Controller: the TMS class.
- Model: other classes.



Design Choice: Inheritance and Polymorphism

- Inheritance: reusability, scalability.



Design Choice: Inheritance and Polymorphism

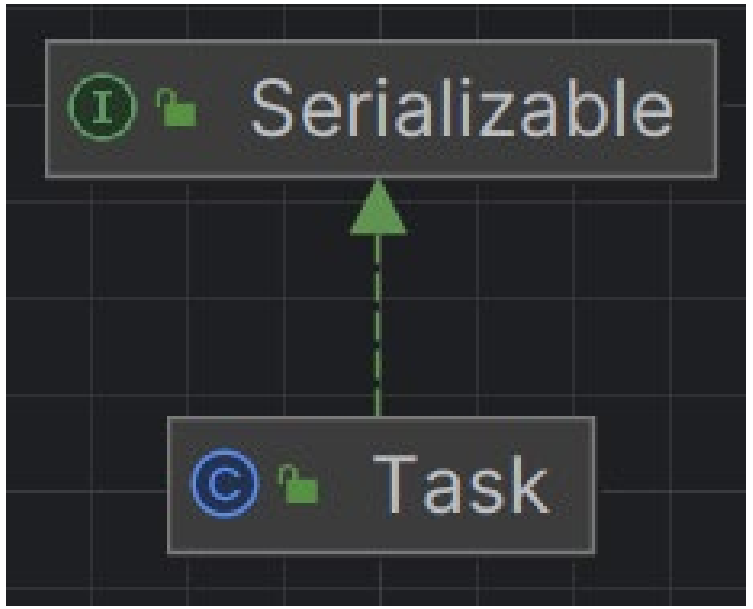
- Polymorphism: flexibility, modularity.
- Example: As long as it evaluates.

```java
/**
 * @param criterion the criterion with which we filter the tasks
 * @return execution message
 */
1 usage  LinZhanzhi
public String search(Criterion criterion) {
    StringBuilder feedback = new StringBuilder();
    ArrayList<Task> desiredTask = new ArrayList<>();
    for (Task task : getTaskCollection())
        if (criterion.evaluate(task))
            desiredTask.add(task);
    feedback.append("Search result: ").append(desiredTask.size()).append(" tasks:\n");
    for (Task task : desiredTask)
        feedback.append(task);
    return feedback.toString();
}
```

Design Choice: How about task?

- composite task is complete the moment its subtasks are completed

- a composite task as a primitive task with a zero duration



Future Enhancements

- Lack of reusability, scalability
  - Inheritance not implemented for Task

Conclusion

Q&A

How about TaskSet and Cri

Acknowledgement

- Group