



ADDIS ABABA
**SCIENCE AND
TECHNOLOGY**
UNIVERSITY
UNIVERSITY FOR INDUSTRY

Common Refactoring Methods

Software Evolution and Maintenance



Introduction to Refactoring

- **Improving the internal structure of code without changing external behavior.**

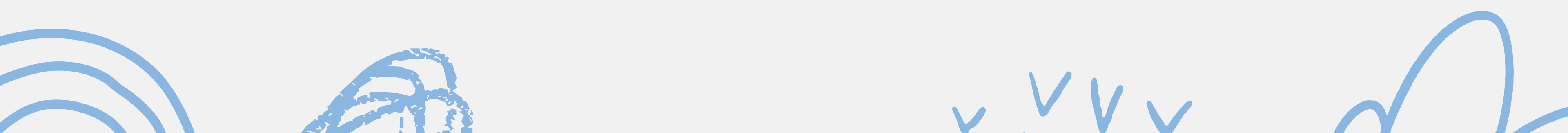
Benefits:

- **Enhances readability and maintainability**
 - **Reduces duplication and improves adaptability**
- 



Common Refactoring Methods

◆	◆	◆	◆	◆
01	02	03	04	05
Extract Method	Inline Method	Move Method	Rename Method	Pull- up/Push- down







Extract Method Refactoring

01. Breaks a large method into smaller, manageable methods

02. Characteristics:

- Improves modularity and readability
- Reduces duplication

03. Example: Splitting a function performing multiple tasks into separate methods







Inline Method Refactoring

01. Replaces method calls with the method's content

Characteristics:

- 02.**
- Simplifies code by removing unnecessary abstraction
 - Reduces method call overhead

03. Example: Directly implementing simple function logic instead of calling it







Move Method Refactoring

01. Moves a method to a more appropriate class

Characteristics:

- 02.**
- Improve class organization and cohesion
 - Reduces dependencies and coupling

03. Example: Shifting a method from a utility class to a domain-specific class





Rename Method Refactoring

01. Changes a method name
for better clarity

Characteristics:

- 02.**
- Enhances readability and developer understanding
 - No impact on performance

03. Example: Changing
"process()" to
"processUserPayment()
"





Pull-up & Push-down Methods

01. Adjusts method hierarchy

Types:

- 02.**
- **Pull-up:** Moves a method to a superclass
 - **Push-down:** Moves a method to a subclass

Benefits:

- 03.**
- **Increases code reuse**
 - **Reduces redundancy**
- 
- 

Refactoring Type	Purpose	Benefits	Drawbacks
Extract Method	Breaks down large methods	Improves readability, reduces duplication	Can create excessive method calls
Inline Method	Replaces method calls with code	Reduces unnecessary abstraction	Can make code harder to modify
Move Method	Moves a method to a better-suited class	Increases cohesion, reduces coupling	Requires careful dependency handling
Rename Method	Changes method name for clarity	Enhances understandability	No direct impact on performance
Pull-up/Push-down	Adjusts method hierarchy	Encourages code reuse	Can disrupt existing class structures

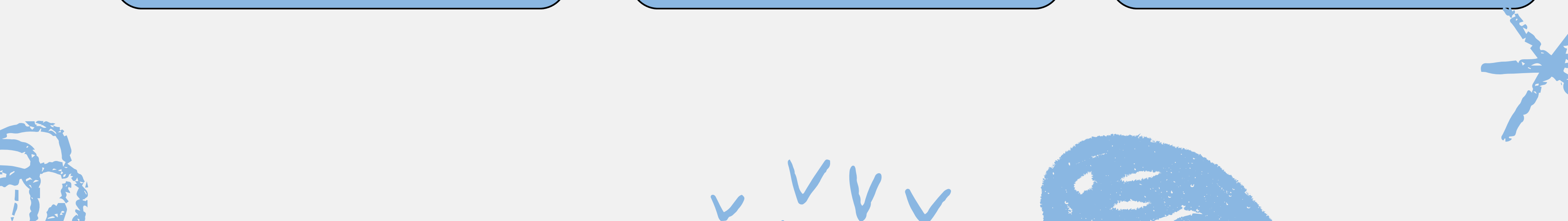


Limitations of Refactoring

Increased Development Time:
Frequent refactoring may
delay new features

Higher Maintenance Costs:
Over-refactoring can lead
to unnecessary complexity

Risk of Bugs:
Improper refactoring
can introduce defects



Conclusion

- Refactoring improves software quality by enhancing readability, modularity, and maintainability
- Choosing the right technique depends on project requirements and complexity
- Automation and AI-driven refactoring tools are emerging trends



The background is a light gray color, decorated with various hand-drawn blue doodles. These include loops, swirls, a zigzag line, and several checkmarks scattered around the central text.

**Thank you
very much!**