Design pattern collection

C1= class “Cfilering”
C2= class “CfilteringDriver”
F1= class “FindRelatedPairOfUsers”
F2= class “FindMostSimilarPairOfUsers”
F3= class “FindMostDissimilarPairOfUsers”
F4= class “FormatPrintMessage”
P1= class “PrintRelatedPairOfUsers”
P2= class “PrintUserUserMatrix”

Handout Given Patterns:

1. ***Single responsibility principle***:
   - Separated multiple methods from original C1 (in A0) into different classes(F2, F3, P2), each class is responsible to a single function.
   - Separated multiple methods from original C2 (in A0) into different classes(PrintMsg and ReadMsg).
2. ***Liskov substitution principle***:
   - F2, F3 are subtypes of F1, any methods written in F1 can work equally well in F2 and F3.
3. **Interfaces**:
   - Created Interface F4, used in P2, F2, F3 which have to print format messages.
4. **Abstract classes**:
   - Created Abstract class F1. For example, F2 is a F1, F3 is a F1.
5. **Unit testing**: All files in test folder.
6. **Exceptions**: All files in exception folder.

Other Design Patterns:

1. **Iterator**:
   - In P1 line50-54 traversal list of user to print them out.
   - Tried to use in C2, but that would be much more lines than using current way in this case.
2. **Singleton**:
   - In C1 line25-30, 67-132.
3. **Builder**:
   - In C2 line44-47
   - In F2 line49
   - In F3 line48
   - In PrintMsg line45-48
   - In ReadMsg line63
4. **Polymorphism**:
   - P1(can print dissimilar and similar etc.) can deal with F1 and F2
5. **Factory method**: Implemented F1 by two types: F2 and F3
6. **Principle of least knowledge**: Each class in driver only to talk to immediate friends, have only limited knowledge about other units.