COMP 3111
SOFTWARE ENGINEERING

LECTURE 11
IMPLEMENTATION
REFACTORING EXERCISE

EXERCISE: REFACTORING A METHOD

A class called `Group` has a method called `getUsers()`. On the exercise sheet are three versions of that method. Rank the versions from best (1) to worst (3) by writing the rank in the box to the left of the version.

From: https://elearning.industriallogic.com/gh/submit?Action=PageAction&album=foundations&path=foundations/refactoringExercises/orderGroupMethodRefactoring&devLanguage=Java

EXERCISE: REFACTORING A METHOD

Version 1

```
public class Group…
    public List getUsers() {
        List users = new ArrayList();

        if (!userDirectoryExists())
            return users;

        // Add found users to users collection
        File[] files = new File(persistencePath()).listFiles();
        for (File file : files)
            if (file.isDirectory())
                users.add(new User(file.getName(), this));

        // Sort by most recently registered users
        Collections.sort(users, new User.UserComparatorByDescendingRegistration());

        return users;
    }
```

EXERCISE: REFACTORING A METHOD

Version 2

```
public class Group…
    public List getUsers() {
    List users = new ArrayList();
        if (!userDirectoryExists())
        return users;
    addFoundUsersTo(users);
    sortByMostRecentlyRegistered(users);
    return users;
    }
```

EXERCISE: REFACTORING A METHOD

Version 3

```
public class Group…
 // Gets users sorted by most recently registered user
   public List getUsers() {
      List users = new ArrayList();
      if (!new File(persistencePath()).exists())
         return users;
      File[] files = new File(persistencePath()).listFiles();
      for (File file : files)
         if (file.isDirectory())
            users.add(new User(file.getName(), this));
      Collections.sort(users, new User.UserComparatorByDescendingRegistration());
      return users;
   }
```

EXERCISE: REFACTORING A METHOD—SOLUTION

Version 2: Rank 1

Reason: It is easy to understand how the method works, and there is no need for comments. Even the method name helps to communicate exactly what kind of data the method returns. This method is a good example of what self-documenting code can be.

Version 1: Rank 2

Reason: This version uses comments to help clarify what the code does, instead of extracting code into well-named methods to communicate what each step of the method does.

Version 3: Rank 3

Reason: The comment at the top is deodorant for code that does not quickly communicate what it does. You have to puzzle your way through the lines of code to figure out what it is doing.