Classroom Exercise:
"CODE REVIEW"

1) Preparation:

a) Advance reading

In order to perform this exercise, you have to read the following papers:

[1] Fagan, M. E. , Advances in Software Inspections, IEEE Transactions on Software Engineering, Vol. 12(7):744-751, July 1986.
[2] Alberto Bacchelli, Christian Bird, "Expectations, outcomes, and challenges of modern code review," In: Proceedings 35th International Conference on Software Engineering (ICSE), IEEE, 2013.: 712-721.

These two papers are the same papers included in the reading list of the lecture of the course "***Peer Code Review: Theory and Practise*"** (held on 20-03-2015).

b) Project

Each student should select a Java project implemented in his/her past academic/professional experience.

2) Background:

Code Review is the "systematic examination by one's peers of computer source code intended to find and fix mistakes overlooked in the initial development phase".

In both industrial and open source projects it is a well-known fact that bugs found later in the design cycle are more expensive to correct than those found earlier. In the worst case, the product ships with bugs that require a maintenance release, costly in-the-field upgrades and possibly can damage the reputation of the company. One of the most cost-effective

ways of eliminating bugs very early in the development cycle is to perform source code reviews. During this activity developers study each other's code with an eye towards finding coding problems. The upfront investment in time and effort saves time and expense later on. Code reviews is performed by developers of popular software companies such as, Google, Microsoft and Facebook.

Code Reviews at Google:
"*All code that gets submitted needs to be reviewed by at least one other person, and either the code writer or the reviewer needs to have readability in that language. Most people use Mondrian to do code reviews, and obviously, we spend a good chunk of our time reviewing code.*"
Amanda Camp, Software Engineer, Google

Figure 1. shows the typical Code Review process (Slides in https://www.olat.uzh.ch/olat/auth/1%3A2%3A5001853181%3A4%3A0%3Aserv%3Ax/slides/SWEvol-FS15-Code_Review.pdf explain more in details this process).

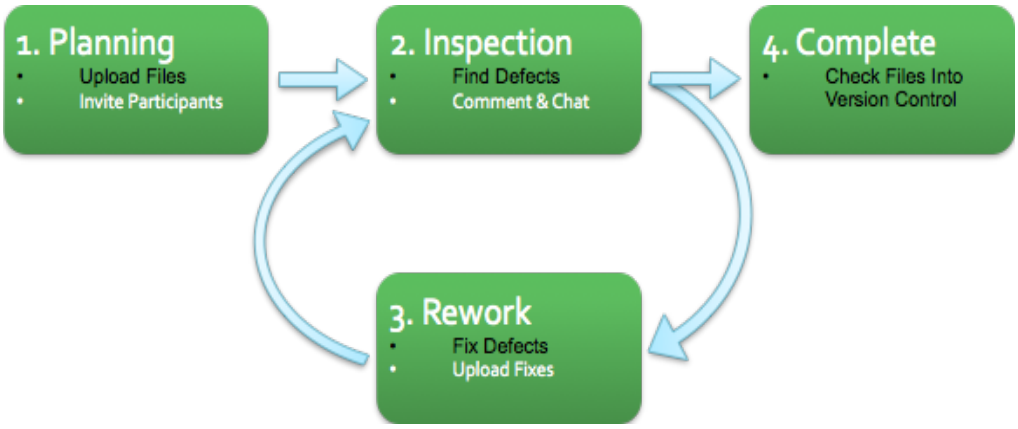

Figure 1. Code Review Process.

Specifically, during code review, developers can play the role of *reviewers* and of *verifiers*. Reviewers have the responsibility to give feedbacks and comments to improve the quality of the proposed patches. In addition, verifiers also evaluate whether the patches are really useful to fix problems/defects without breaking the behavior of the system. Thus, if patches meet the defined criteria, they are automatically integrated into the master repository and their status is changed to merged.

3) A Simple Example of Code Review (from the Book of "*Applied Software Project Management*" by Andrew Stellman and Jennifer Greene):

```
public class Account {
double principal,rate; int daysActive,accountType;
public static final int STANDARD=0, BUDGET=1, PREMIUM=2,
PREMIUM_PLUS=3;
}
...
public static double calculateFee(Account[] accounts)
{
double totalFee = 0.0;
Account account;
for (int i=0;i<accounts.length;i++) {
 account=accounts[i];
        if(account.accountType==Account.PREMIUM|| account.accountType
== Account.PREMIUM_PLUS )
        totalFee += .0125 * ( // 1.25% broker's fee
        account.principal*Math.pow
        (account.rate,(account.daysActive/365.25))
        - account.principal); // interest-principal
}
return totalFee;
}
```

Improved code:

```
/** An individual account. Also see CorporateAccount. */
public class Account {
     private double principal;
    /** The yearly, compounded rate (at 365.25 days per year). */
   private double rate;
   /** Days since last interest payout. */
   private int daysActive;
   private Type type;
   /** The varieties of account our bank offers. */
   public enum Type{STANDARD, BUDGET, PREMIUM,     PREMIUM_PLUS}
   /** Compute interest. **/
public double interest() {
    double years = daysActive / 365.25;
   double compoundInterest = principal * Math.pow(rate, years);
   return compoundInterest – principal;
}
```

```
/** Return true if this is a premium account. **/
public boolean isPremium() {
      return accountType == Type.PREMIUM ||
      accountType == Type.PREMIUM_PLUS;
}

/** The portion of the interest that goes to the broker. **/
public static final double BROKER_FEE_PERCENT = 0.0125;
/** Return the sum of the broker fees for all the given accounts. **/
public static double calculateFee(Account accounts[]) {
   double totalFee = 0.0;
      for (Account account : accounts) {
         If (account.isPremium()) {
            totalFee += BROKER_FEE_PERCENT * account.interest();
    }
 }
return totalFee;
}
}
```

Problems and defects discovered and removed:

1. Many comments are missing or are not informative enough
2. The indentation style is not consistent and makes it more difficult to read the code.
3. Some variables look like class names.
4. Some comments are misleading.
5. Many methods are too large and should be broken into smaller methods.
6. Some variables are never used.
7. Perhaps regular expressions should be assigned to strings to improve the readability.

4) Exercise:

The goal of this exercise is to review, discuss, and provide constructive criticism about code a peer has developed. In the real world we will need to read/understand other people's code, therefore this is a skill we need to develop.

Please, follow the **steps** provided below while completing this exercise:

1. Each student should select a Java project implemented in his/her past academic/professional experience.

2. A *team* of students (composed by at least 3 members) should exchange the code to each of the *member* of the *team*.
3. Each member of a team should asks the other members of the team to review his/her code (***1 or 2 java classes***):
    a. Each member should explain each of the method he/she implemented.
    b. ***Provide a list of aspects to improve relying on a specific check list*** similar to the following:
    i) https://www.liberty.edu/media/1414/%5B6401%5Dcode_review_checklist.pdf
    ii) http://courses.cs.washington.edu/courses/cse403/12wi/sections/12wi_code_review_checklist.pdf
    Examples of criteria to use can be:
        i. Code Clarity
        ii. Variable names
        iii. Indentation
        iv. Do you feel you can modify the code easily?
        v. What is your opinion about the style used in Java Constructs (e.g., loops, methods, etc.)
    c. As the discussion progresses, suggest ways in which the implementation should be modified/improved.
    d. The students must to use PDM (https://pmd.github.io/) and Checkstyle (http://checkstyle.sourceforge.net/) for automatic the finding of some of the relevant warnings.

4. Each Student ***should write a report*** (***max 2 pages***) that include:
    a. Project Description: a high level description of the project under review (specifying the classes selected);
    b. *Team*: the list of the team members;
    c. *Defects Found:* A description of each fault found by each team member;
    d. *Summary of the Recommendation:* a summary of the team recommendations to improve the code.
    e. *Review Time and Defects Found*: a summary of the effectiveness of each review. Specifically, the amount of time that each reviewer spent reviewing the code and the number of defects that each member identified.
5. Together with the report each student ***should attach the two versions of the reviewed class*** (before and after the reviewing process).