Project Deliverables for Release 1
****

The goal is to do professional work and produce deliverables with high quality while observing the third principle in the IEEE-CS/ACM Software Engineering Code of Ethics.

Third principle in the IEEE-CS/ACM Software Engineering Code of Ethics:

"Software engineers shall ensure that their products and related modifications meet the highest professional standards possible."

I will not accept anything submitted via email.

Each team will have to do a demo of their release to the customer (instructor).

Submit a three-hole binder (not one of those 3-ring hardcover binders - hard to carry) with the following contents. Separate each part with a labelled separator.

The binder must contain the following in the order specified below:

1. Cover page with project title and team member names
2. Project objectives (vision and scope)
3. Use case tool reports
4. Class diagram(s) generated from ObjectAid UML Explorer
5. Checkstyle violations charts (before and after modifications charts)
6. Eclipse Metrics reports
7. Git log (to see activity of members with the code repository)
8. Source code
9. CodeCover reports
10. Role and Responsibilities of each member in the team
11. Self-reflection by team
    - release planning issues, team-oriented issues, acquiring skills needed
    - obstacles faced and strategies and techniques used
    - lessons learned
    - any other information you may wish to add
12. Other optional parts (such as JUnit tests, user's guide, etc.)

Version Control with Git
****

Display the history of changes to the git repository.

Use "git log" command to get a listing of what changes have been made to the repository.

```
$ git log
```

Save the output of this command to a file and include this in the report.

Or, use this command to get the history with information in a concise format

```
$ git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short
```

Save the output of this command to a file and include this in the report.

UCTool (Use Case Tool)
****

Prepare use cases
- one or more xml files containing use cases (see sample.xml)

Select Run -> Run Configurations...
Click on "Maven Build"
Press New button at the top to create a new Run configuration
Give a name to the configuration
In Main tab, enter "clean generate-resources" for "Goals:"
Press Run

UCTool runs and generates the "target" folder (press F5 to refresh)
Find index.html in the target/site/* folder
Open index.html file
What to print:
index.html
Actors overview page (click on "Actors" in index.html)
Actor details page (one for each actor from the "Actors overview" page)
Use Cases overview page (click on "Use cases" in index.html)
Use case details page (one for each use case from the "Use cases overview" page)

Checkstyle
****

(One Time) Create custom checkstyle configuration for this course using
the steps below:

Eclipse -> Preferences -> Checkstyle
Select "Sun Checks (Eclipse)" under "Global Check Configurations"
Select "Copy..."
Use "CIS350 Sun Checks (Eclipse)" for Name
Press OK

Select "CIS350 Sun Checks (Eclipse)" under "Global Check Configurations"
Select "Set as Default"
Select "Configure..."
Select "Suppression Filter" under "Filters" under "Known Modules"
Press "Add...->"
Double click "Suppression Filter" in the right pane
Browse to select the supplied file "CIS350-checkstyle-suppressions.xml"
Press OK (3 times)

Enable Checkstyle for the project

Project -> Properties
Select Checkstyle in the left pane
select checkbox for "Checkstyle active for this project"
Press OK
Checkstyle will start check for violations

Submit Before and After Checkstyle Violations Chart

Window -> Show View -> Other... -> Checkstyle -> Checkstyle violations chart
Right click in the chart window and select "Save as..."

Save the chart to a file
Print the image file showing code violations before making any changes

Make changes to code to resolve any violations

Print checkstyle violations chart after making changes. If you fixed all code violations reported by checkstyle, the chart will be empty (no image to save!)

Disable Checkstyle for the project

Project -> Properties
Select Checkstyle in the left pane
Deselect checkbox for "Checkstyle active for this project"
Press OK

Eclipse Metrics
****

- Enable Metrics Gathering for the project
  (Properties -> Enable Metrics Gathering)

- Set Cyclomatic Complexity upper bound to 8
  (Preferences -> Metrics tab)

- Export metrics data
  Select Project/Export...
  Create "MetricsReport" directory in the project folder
  The plugin will generate reports in this folder

- Refresh the Project (Press F5) - You should see "MetricsReport" folder

- Open the index.html file in the MetricsReport folder

- Turn in the following four reports:
  1. Visual Summary (dashboard.html)
  2. Order by Package under "Package Metrics"
  3. Order by Qualified Type Name under "Type Metrics"
  4. Order by Qualified Type Name under "Methode Metrics"

  Bonus: MetricsReport/src folder has *.html files for your source files that you can open and print.

- Disable Metrics Gathering for the project
  (Properties -> Enable Metrics Gathering)

ObjectAid UML Explorer
****

To create a class diagram using the ObjectAid UML Explorer:

Select project -> New -> Other... -> ObjectAid UML Diagram -> Class Diagram
Pick a name for class diagram (for Name text field)
Enable "Save image with Diagram as"

Enable "Add Dependencies"

```
        Enable "Always Add Relationships"
        Disable "Show Association Multiplicity"
        Disable "Show Association Labels"
        Click Finish

        Drag and drop classes/interfaces on to the diagram window

        Right click on the window, select Options, deselect "Show Icons"
            (this will replace the icons and colors in the diagram with standard
             UML symbols for modifiers public, private, and protected, etc. Useful
             for printing a class diagram in black and white)

        Save the diagram and print the saved image file.

        Note: It is possible to have more than one class diagram if there are too
        many to fit on one page. (make perfect sense to have one class diagram per
        package or subsystem)


CodeCover
*********

    CodeCover Eclipse Plugin Reference Manual
    http://codecover.org/documentation/references/eclManual.html

    Enable CodeCover
        Properties -> CodeCover -> Check Enable CodeCover

    Select packages and/or classes to instrument for coverage metrics
        Right a package or class and select "Use For Coverage Measurement"

    Conduct system tests using the Live Notification feature of CodeCover (see manual)

    Generate coverage reports using the following steps:

        Right click Project and select "Export..."
        Select "Coverage Result Export" under CodeCover
        Press Next

        Select a container under "Test Session Container"
        Select All under "Available Test Sessions"
        Select "Report" under "Type"
        Click "Browse...":
            I recommend you create a subfolder in your project named "CodeCoverReport"
            Give a name to the file such as "report" (will be saved as report.xml)
        Press Next

        For "Report Template", browse and select the file HTML_Report_hierarchic.xml in
            /eclipse/plugins/org.codecover.report.html_1.0.1.2/report-templates/
        Press Finish

        Press F5 to refresh the Project
        You should now see a folder named "CodeCoverReport"
        Within this folder, find and print the report (index.html) for each class
```