# How I Design and Write a Batch Job in Java

Steps)

1) Analysis – What does this job need to do?
    a. Output(s) – outputs are not only files, but DB updates, and temp results. What changes in this world after the job has finished?
        i. You need the layout of all outputs.
    b. Input(s) – where do I get the data to build the output(s)
        i. Read a file, XML, DB, command line params, etc.
        ii. Match all the fields on the output(s) to an input
2) Design – How many classes do I need to create?
    a. Main entry point – Application Class
    b. A Class for each set of data – Entity Class
    c. A Class for each input/output – Action Class
    d. Utility Classes – common tasks
        i. Custom Exceptions
        ii. Properties readers
        iii. Sort/Merge/etc.
    e. 3rd party frameworks
        i. Apache Commons
        ii. DB2 Driver for DB access
        iii. Junit for testing
        iv. Etc.
3) Coding – I will now write the code
    a. Start with the Main class
        i. I will outline the program flow
            1. Use the Eclipse wizards to generate the classes and methods accessed within main()
    b. Then write the Entity class(es)
        i. Private fields
        ii. Get/Set
        iii. Constuctor(s)

iv. toString
v. hashCode/equals
vi. JavaDocs
vii. Unit Tests
viii. Log statements
ix. Etc.

c. Then write the Action class(es)
    i. The Interface
    ii. Then the Implementation
        1. Check incoming params
        2. Check permission
        3. JavaDocs
        4. Log statements
        5. Unit tests
        6. Etc.

4) Deployment – Moving the code to the Mainframe
    a. Build a place on the mainframe to hold your deployment
    b. Copy the 3rd party jar file to the mainframe
    c. Export my code as a Remote Jar file
        i. Set the classpath
            1. The exported jar
            2. 3rd party jars
        ii. Set the Arguments
        iii. Set the main class
        iv. Set the working directory
    d. Debug/Run configuration
        i. Run the code – Integration test (starter)
        ii. Check the output(s)