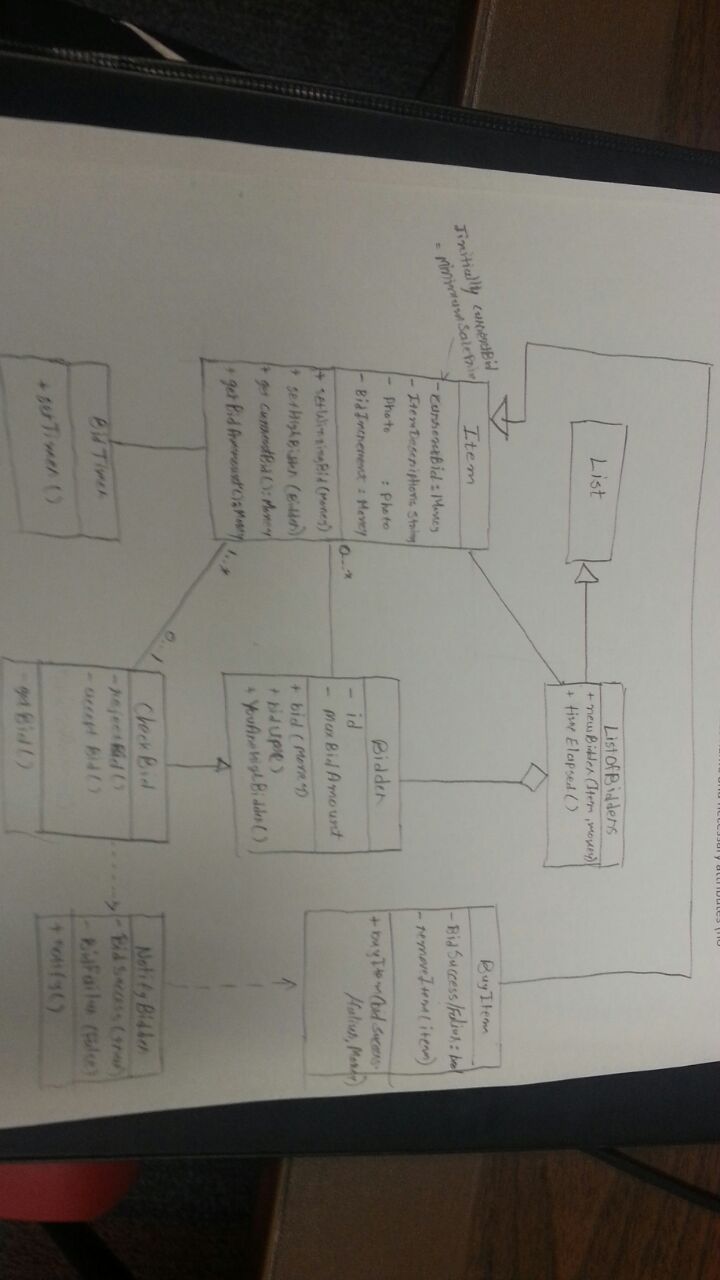
UML – 25 points:

You are modeling an online marketplace (such as eBay) in which sellers put item up for sale in auction format, with a minimum sale price, and the date and time of closing the auction. For each item on auction, interested buyers will place bids on the item, and at the closing time of the auction, the item will be sold to the person who has bid the highest price.

Draw a UML diagram for your model. Identify classes clearly, and separate them accordingly to functionality – the separation should be faithful to the description above. Set up relationships between classes as precisely as possible-for instance, if a relationship is an aggregation, depicting it as a generic association is not precise enough. For each class, only list the class name and necessary attributes (no operations needed)



Testing – 25 points:

A method groups an array-list of music tunes into the categories rock, R&B, and Jazz, such that all tunes in a category are contiguously placed in the list. Within any category, the tunes can be in any order, and the categories themselves can be in any order. Treating the method as a black box (i.e.implement in not visable/accessible), list test-cases to check its correctness. Make sure you detail the equivalence classes, and the tests for each equivalence class.

Multithreading – 25 points:

1. (15 – points) Write a program called WordCount that counts the number of words in one or more files. Start a new thread for each file. Here is sample run:

* Java WordCount report.txt address.txt Homework.java

Address.txt: 1052

Homework.txt: 2099

Total: 3595

You may assume that a word is any sequence of non-white space characters. The outputs need not be in any perticulat order. Also, you can assume the following method has been implemented fo you:

Public static int countWords(String filename) {. . . . }

* **Public class** WordCount{

**public** **static** **void** main(String args[])**throws** InterruptedException

{

**for** (String filename : args)

{

Runnable tester = **new** WordCount(filename);

Thread t = **new** Thread(tester);

t.start();

}

}

**public** **void** run(){

**this**.CountWords(filename);

}

}

1. (10 - points) Suppose there is a class called Database that holds a private ArrayList of (thousands of) User objects. Each User object has several data field including username, password, email, etc. All access to the database are multithreaded, so that at any given time each of several threads may be trying to read or update the User instances. Two threads will conflict is one of them is trying to write to a user instance at the same time as another is trying to read the same instance. However, there is no conflict if different User instances are accessed at the same time by different threads. Show (with relevant code segment) how you will make access to User instances via the Database object thread-safe while maintaining concurrency.

* Java enables to coordinate the actions of multiple threads by using Synchronized Methods and Synchronized Statements.
* Here we can access User Object with Synchronized methods. Only one synchronized method can be invoked for an object at a given point in time, which keeps Synchronized methods in multiple threads from conflicting each other.
* OR we can simply change the ArrayList of objects to List of objects and use collections.synchronizedList().

private List Users;

Users = collections.synchronizedList(new ArrayList(numberOfUsers))

Design Patterns – 25 points:

You are developing an application for an online travel agency. The agency manages several 5-day packages to various tourist sites. Each package consist of transport to site (day 1), followed by activities for each of the 3 days, and finally transport back home( day 5). The transport details and the actual activities differ depending on the package.

Show how you would use the template method design-pattern to model the various tour packages, When a user picks a package, the application should print the entire sequence of activities for that package. Sketch the appropriate classes and methods, with sufficient detail to clearly demonstrate the design pattern.

* **public** **class** Trip {

**public** **final** **void** performTrip(){

TransportToSiteDay1();

Day2();

Day3();

Day4();

TransportBackHomeDay5();

}

**public** **abstract** **void** TransportToSiteDay1();

**public** **abstract** **void** Day1();

**public** **abstract** **void** Day2();

**public** **abstract** **void** Day3();

**public** **abstract** **void** ();

}

**public** **class** PackageA **extends** Trip {

**public** **void** TransportToSiteDay1() {

System.*out*.println("comming by air ...");

}

**public** **void** Day2() {

System.*out*.println("visiting the aquarium ...");

}

**public** **void** Day3() {

System.*out*.println("going to the beach ...");

}

**public** **void** Day4() {

System.*out*.println("going to mountains ...");

}

**public** **void** TransportBackHomeDay5() {

System.*out*.println("going home by air ...");

}

}

**public** **class** PackageB **extends** Trip {

**public** **void** TransportToSiteDay1() {

System.*out*.println("comming by train ...");

}

**public** **void** Day2() {

System.*out*.println("visiting the mountain ...");

}

**public** **void** Day3() {

System.*out*.println("going to the beach ...");

}

**public** **void** Day4() {

System.*out*.println("going to zoo ...");

}

**public** **void** TransportBackHomeDay5() {

System.*out*.println("going home by train ...");

}}