OOP Term Project Ideas

Goal: To learn how to build and evolve large-scale programs using object-oriented programming, and work in teams learning from each other.

Topics: In exploring object-oriented programming, we investigate three questions:

- **Design**: How do we think about a program in terms of objects? To answer this question, we explore CRC cards, UML, and design patterns.
- **Primitives**: How do we express object orientation? To answer this question, we explore classes, interfaces, inheritance, method dispatch, generics, operator overloading, and reflection.
- **Implementation**: How do we realize object-oriented primitives? To answer this question, we explore virtual method dispatch and automatic memory management in detail.

Design and primitives matter because they represent the essence of object-oriented programming. Implementation matters because it enables us to debug object-oriented programs and tune their performance.

General Rules:

- A group can be formed from 3 or 4 students per project.
- Groups are not allowed to repeat ideas, so first come first serve bases, once a group comment on the project's thread with a choice, the idea can not be chosen by another group.

Grading Criteria:

The project is worth10% of your final mark. These marks are graded individually by asking each student in the presentation for their contribution and testing their understanding. These are broken into:

- 5 marks for correctness (no compilation or run-time errors).
- 4 marks for the application of course concepts, where feasible, (such as: modularity, inheritance, polymorphism, method overriding and overloading, abstract classes & interfaces)
- 1 marks for GUI interfaces, presentation, documentation, and teamwork.
- 5 bonus marks for all other concepts you self-study outside the learning objectives of the course.

Submission Details:

All project files are zipped and submitted named as "proj_LeaderStudentID.zip", where "LeaderStudentID" is replaced by a leader team member chosen by all team members.

The zip file should contain:

1. All source code used to develop the project, either as a jar file, or a folder of the project packages and files.
2. A Design document containing a class diagram describing the project classes, and associations, and one paragraph describing the methods, one paragraph as a user manual, and one paragraph describing team members individual contributions.

Hints:

Below are project ideas and sample simplified requirements. Please elaborate on classes, attributes, methods as you need, and be creative. Writing Data to tab delimited files are important to manage data properly and be able to read it back. You can update the following functions to suit the requirements of your projects.

```
public static boolean WriteToTabDelimitedFile(String[] data, String FileName) {
   try {
        FileWriter fileWritter = new FileWriter(FileName);
        BufferedWriter bufferWritter = new BufferedWriter(fileWritter);
        // loop through all your data and print it to the file
        for (int i=0;i< data.length;i++)
               bufferWritter.write(data[i]+"\t");
        bufferWritter.write("\n");
        bufferWritter.close();
   } catch (IOException e) {
        System.out.println("Error Printing Tab Delimited File");
        return false;
   }
   return true;
}
public static boolean appendToTabDelimitedFile (String[] data, String FileName) {
   try{
        File file =new File(FileName);
        if(!file.exists())   //if file doesn't exists, return false
               return false;
        FileWriter fileWritter = new FileWriter(file.getName(),true);
        BufferedWriter bufferWritter = new BufferedWriter(fileWritter);
        for (int i=0;i< data.length;i++)
               bufferWritter.write(data[i]+"\t");
        bufferWritter.write("\n");
        bufferWritter.close();
   } catch (IOException e){
        e.printStackTrace();
   }
   return true;
}
public static String [] readFromTabDelimitedFile (String FileName) {
   List<String> lines = new ArrayList<String>();
   try {
        FileReader fileReader = new FileReader(FileName);
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        String line = null;
        while ((line = bufferedReader.readLine()) != null)
               lines.add(line);
        bufferedReader.close();
   } catch (IOException e) {
        e.printStackTrace();
   }
   return lines.toArray(new String[lines.size()]);
}
public static void main (String[] args) {
   // Example Use of the above functions
   String[] data= {"Course", "Lecturer","TA"};
   WriteToTabDelimitedFile(data, "courses.txt");
   data[0] = "CS243";data[1] = "Manal";data[2] = "Hend";
   appendToTabDelimitedFile(data, "courses.txt");
   data[0] = "CS244";data[1] = "Manal";data[2] = "Maiada";
   appendToTabDelimitedFile(data, "courses.txt");
   String[] readData = readFromTabDelimitedFile("courses.txt");
   for (int i = 0;i<readData.length;i++) {
        String [] tokens = readData[i].split("\t");
        for (int j = 0;j<tokens.length;j++)
           System.out.print(tokens[j] + "\t");
        System.out.println();
   }
}
```

Project Ideas (first 6 and data management projects, then GUI games, then animation projects):

1) Bank Management System:

Develop an application to help a bank manager manage customer accounts. The bank offer several bank accounts types. Each customer can have one or more accounts. The customer can go the operations permitted by the account type, such as deposit, withdraw, or balance enquire. The bank manages the account by debiting the fees, or crediting the profits. Both the bank employees and the customers can print reports about the current account details.

Design:
Basic Classes: Account, CheckingAccount, SavingAccount, Loan, Customer:

A. The Account is a general account class that contains balance as instance variable, deposit, withdraw, and balanceEnquiry as instance methods.
B. CheckingAccount is a subclass from the Account class that allows overdraft while withdrawing (making the balance go below zero up to the specified credit limit), by debiting the account balance with an overdraft fee . It has a creditlimit as an instance variable.
C. The Saving Account is a subclass from the Account class that has an interest rate as an instance variable. The system credit the balance with monthly interest based on the account balance and the interest rate.
D. The loan account is a subclass from the Account class that has principal amount, interest rate, loan duration in months as instance variables. The loan balance is debited by monthly interest each month based on the interest rate and the loan balance.
E. Each customer can have any number of accounts of any type.
F. The banking system provide the customer with an interface to access all banking operations described above and review reports about transactions and current balance.
G. A banking administrator can print a report about all customers and their current balances.

Provides an interface for the user to:

1. Adding/editing/deleting GUI to each class,
2. GUI for Customers to open a new account
3. GUI for Customers to view transactions and balance for all their accounts.
4. GUI for a administrator to view all customer balances.

Sample data include:
customers.txt

Customer ID	Name	Address	Phone	Email
1	Mohamed	…	…	…
2	Ahmed	…	…	…
3	Mostafa	…	…	…

accounts.txt

Account ID	Customer ID	Type	Balance	CreditLimit or Interest Rate	Principal Amount	Loan Duration
1	1	Saving	…			
2	1	Loan	…			
3	2	Checking Account	…			

accountTransactions.txt

Account ID	Date Time	Transaction Type	Amount	Transaction Type
1	9/1/2013	Withdraw	…	Withdraw
1	…	Deposit	…	Rowing machine
2	…	Interest	…	Ab Rolller
3	…	Fees	…	

2) Gym Management System

Develop an application to help a gym manager manage a GYM hall, with various equipments, trainers/customers, equipment's, and exercise plan, and simple scheduling of the customers subscriptions.

Design:

Basic Classes: Gym halls, equipments, trainers, exercise plans and customers:

H. The Gym hall aggregates several equipments.
I. Each hall opens 12 hours a day.
J. Each trainer works 8 hours a day in one hall.
K. A trainer submits a daily exercise plan with steps assigned to each equipment for a specified duration in minutes.
L. Each customer subscribe to a particular hall, at a particular time and date, with a particular trainer, with a specified exercise plan for one month.

Provides an interface for the user to:

5. Adding/editing/deleting GUI to each class,
6. The system should check the availability of trainer and the equipments listed in the exercise plan at the time chosen by the customer, before adding a new subscription.
7. Enquire about customers subscriptions, halls schedule, and trainers schedule.

Sample data include:

customers.txt

Customer ID	Name	Address	Phone	Email
1	Mohamed	…	…	…
2	Ahmed	…	…	…
3	Mostafa	…	…	…

trainers.txt

Trainer ID	Name
1	Mohamed Ahmed
2	Mahmoud Ali
3	Sami Selim

equipments.txt

Equipment ID	Name
1	Treadmill
2	Rowing machine
3	Ab Rolller

excercisePlans.txt

Plan ID	Trainer ID	Equipment ID	Duration
1	1	1	5
1	1	2	20
2	2	3	Coffee Mug

subscriptions.txt

Subscription ID	Date/Time	Customer ID	Trainer ID	Exercise ID
1	9/1/2013	1	1	6
1	9/1/2013	1	2	1
2	12/2/2013	3	3	2
2	13/2/2013	3	…	…

3) Document Management System

Develop an application to manage document storage and retrieval.

Design:

Basic Classes: Category, Document, Topic, Tag, and have the following relationships:

A. A document belongs to a category such as policy, plan, report, receipt, order, … etc.
B. A document belong to a topic such "CS243 Course Files in Fall 2013", "Cluster Graduation Project in 2013", … etc.
C. A document can have any number of tags such as: "legal", "medical", "administrative", "technical", "2013", "reporting", … etc.

Provides an interface for the user to:

A. Adding/editing/deleting instances belonging to each class,
B. Retrieve document by Category, Topic, Tag.

Sample data include:

category.txt

Category ID	Name
1	Policies
2	Plans
3	Reports

topic.txt

Topic ID	Topic	Storage Folder
1	CS243 Course Files in Fall 2013	C:\Users\Username\Documents\CS243_F2013\
2	Cluster Graduation Project in 2013	…

document.txt

Document ID	Category ID	Topic ID	Tags	Filename
1	2	1	[Syllabus; Educational]	CourseDescription.pdf
2	1	…	…	…
3	2	…	…	…
4	2	…	…	…

4) Library Management System

Develop an application to manage Library borrowing and returns activities.

Design:

Basic Classes: Library, Contents, Books, Articles, Digital Media, Subscribers, Borrowing Records, and have the following relationships:

A. A library content can be a book, an article in a journal, or digital media.
B. A subscriber can borrow any of the library contents, for three weeks, golden subscribers can borrow for three months, after the lease time is over, a fee accrue on the subscriber until the borrowed item is returned.

Provides an interface for the user to:

A. Adding/editing/deleting instances belonging to each class,
B. Subscriber browsing library contents and select items to borrow.
C. Subscriber returning borrowed item and check his balance and pay any late fee if any.
D. Admin can print reports of overdue borrowed items.

Sample data include:

library.txt

Library ID	Name
1	Main Campus Library
2	CS Library
3	Engineering Library

items.txt

Item ID	Library ID	Category	Title	Author	Publisher	Production Year	Status	Copies
1	1	book	Introduction to Java	Daniel Liang	Pearson	2010	On shelf	10
2	…	…	…	…	…	…	…	…

subscribers.txt

Subscriber ID	Type	Name	Address	Phone	Email
1	regular	Mohamed	…	…	…
2	golden	Ahmed	…	…	…
3	regular	Mostafa	…	…	…

Borrowing.txt

subscriber ID	Borrow Date	content ID	Return Date	Fee
1	10/1/2013	1	…	0
1	10/1/2013	1	…	50
2	12/2/2013	3	…	…
2	13/2/2013	3	…	…

5) Project Management System

Develop an application to manage Project tasks and members.

Design:
Basic Classes: Projects, Team Members, Tasks, Resources, and have the following relationships:

A. A project can have many tasks.
B. A task can be done my several team members and require several resources. A team member can be involved in several tasks at the same time, but the resource should be reserved for only one
A. task at a time.

Provides an interface for the user to:

A. Adding/editing/deleting instances belonging to each class,
B. Define tasks, and log progress and completion.
C. Print reports about delayed tasks.

Sample data include:

project.txt

Project ID	Name
1	Term Project
2	Wedding Plans
3	…

members.txt

Member ID	Name	Address	Phone	Email
1	Mohamed	…	…	…
2	Ahmed	…	…	…
3	Mostafa	…	…	…

Resources.txt

Resource ID	Name
1	Laptop 1
2	Printer 1
3	…

tasks.txt

Task ID	Project ID	Title	From Date	To Date	Member ID	Resources	Status
1	1	Design the class UML	10/1/2013	10/1/2013	[1, 2]	[1]	done
2	1	…	10/1/2013	10/1/2013	…	…	…
3	3	…	12/2/2013	12/2/2013	…	…	…
4	3	…	13/2/2013	13/2/2013	…	…	…

6) GUI Projects: Brick Breaking Game

Develop an application to allow a user to play brick breaking.

Requirements:

The game has a grid of bricks of different colours. The aim of the Breaking Bricks game is to empty the entire grid by clicking on bricks of the same colour grouped together. When players click on a group of bricks of the same colour, those bricks disappear and the bricks around them collapse and fill the spaces left by them. If players need to remove only a single brick, they must use their magic wand. The game ends when players use up all their five magic wands.

Bricks breaking games require that players strategize to ensure that they break bricks of the same color grouped together and avoid using their magic wand. Planning the next move so that you are not left with only one brick of a particular colour will ensure that you are left with more magic wands. After completing the first grid, players move on to the next grid and so on, till they run out of magic wands.

Basic Classes: GUI of Bricks arrangements of random colours, User Events of clicking a brick, game logic of identifying groups of same colour to collapse them, and rearrange the screen, game status and ending logic.

Example:
http://www.knowledgeadventure.com/games/bricks-breaking/

7) GUI Projects: Bouncing Balls Game

Develop an application to allow a user to play Bouncing Balls.

Requirements:

Bouncing Balls is a fun game for young kids. The game involves a group of colored balls that move towards the bottom of the screen. The player must destroy all the balls before they reach the bottom. Younger players may play the game without a real strategy in mind, but older kids who play Bouncing Balls must think strategically in order to get a high score. Online games like Bouncing Balls are popular across all age groups. While they do not have any obvious educational value, these games contribute to the cognitive development of the player. The online game Bouncing Balls also has another important aspect to it - the use of principles in physics to play the game. As the balls move towards the bottom of the screen, it becomes more and more difficult to aim directly towards the spot where the ball must land. In these cases, the player must shoot the ball towards the walls on either side and have it bounce back towards the desired spot. The successful execution of this process requires that the player take into consideration basic concepts in physics.

Basic Classes: GUI of balls arrangements of random colours, User Events of clicking a ball, game logic of identifying groups of same colour to collapse them, and rearrange the screen, game status and ending logic.

Example:
http://www.knowledgeadventure.com/games/bouncing-balls/

8) Animation Projects: Computer Generated Animation Images

Develop an application to display artistic types of animations and images, such as fractals, arabesque, geometric shapes, … etc. Use the examples presented in the book and develop them further.